

UNIVERSITY OF CALIFORNIA
Los Angeles

**A Hierarchical and Contextual Model for Learning and
Recognizing Highly Variant Visual Categories**

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Statistics

by

Jacob Matthew Porway

2010

Report Documentation Page		Form Approved OMB No. 0704-0188
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.		
1. REPORT DATE 2010	2. REPORT TYPE	3. DATES COVERED 00-00-2010 to 00-00-2010
4. TITLE AND SUBTITLE A Hierarchical and Contextual Model for Learning and Recognizing Highly Variant Visual Categories		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER	
	5e. TASK NUMBER	
	5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of California, Los Angeles, Los Angeles, CA, 90095		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited		
13. SUPPLEMENTARY NOTES		
14. ABSTRACT <p>In this dissertation we present a hierarchical and contextual model for representing image patterns (manmade objects and aerial images) that are highly variant from instance to instance. These types of patterns are difficult to model because objects within the same class may have very different photometric and geometric properties and/or compositions of parts, e.g. teapots may have very different colors, shapes, and locations of their spouts and handles. We hypothesize that these varied visual patterns can be captured by using a novel representation that arranges common primitives of the patterns in a probabilistic hierarchy, thus compactly capturing possible compositional variations, and then enforces contextual constraints on the appearances of the parts, thus modeling the conditional photometric and geometric relationships of the object parts. We combine a Stochastic Context Free Grammar (SCFG), which captures the long-range compositional variations of a pattern, with a Markov Random Field (MRF), which captures the short-range constraints between neighboring pattern primitives, to create our model. We also present a minimax entropy framework for automatically learning which contextual constraints are most relevant for modeling a type of pattern and estimating their parameters. Finally, we present a novel Markov Chain Monte Carlo (MCMC) algorithm called Clustering Cooperative and Competitive Constraints (C4) for efficiently performing Bayesian inference with our model. C4 is a method for minimizing energy functions defined on graphs that we will use to combine bottom-up and top-down information to find the best interpretation of an image. We show experiments on learning models of a number of manmade object categories and of aerial images and demonstrate that our algorithms automatically learn models that accurately capture the statistical nature of the patterns we are modeling. We also show that our model can be used for inference in new images, allowing it to identify objects in challenging scenarios.</p>		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT Same as Report (SAR)	18. NUMBER OF PAGES 169	19a. NAME OF RESPONSIBLE PERSON
a. REPORT unclassified	b. ABSTRACT unclassified	c. THIS PAGE unclassified			

© Copyright by
Jacob Matthew Porway
2010

The dissertation of Jacob Matthew Porway is approved.

Ying Nian Wu

Alan Yuille

Demetri Terzopoulos

Song Chun Zhu, Committee Chair

University of California, Los Angeles

2010

*To my parents,
who taught me never to pass up an opportunity to learn something.*

TABLE OF CONTENTS

1	Introduction	1
1.1	Motivation	1
1.2	Our Approach and Technical Contributions	4
1.3	Overview of the Dissertation	6
2	Related Work	8
2.1	Appearance-Based Methods	8
2.2	Geometric-Based Methods	10
2.3	Grammar/Compositional-Based Methods	12
2.4	Domain-specific Methods	17
3	The Hierarchical Contextual Model	19
3.1	Stochastic Context Free Grammars	19
3.2	Markov Random Fields	23
3.3	Creating a Contextual Hierarchy	25
3.4	Mathematical Formulation for the Contextual Hierarchy	30
3.5	Application to Object Modeling	33
3.6	Application to Aerial Image Modeling	35
4	Learning via Minimax Entropy	39
4.1	Maximum Likelihood Estimation	39
4.2	Learning $(\lambda^{(\alpha)}, \lambda^{(\beta)})$	40

4.3	Relationship Pursuit	44
4.4	Summary of Parameter Learning and Relationship Pursuit Algorithms	47
5	Experiments on Learning and Sampling	50
5.1	Experiments on Object Learning	50
5.1.1	Data Collection	50
5.1.2	Object Representation	51
5.1.3	Relationship Functions	54
5.1.4	Parse Graph Construction	54
5.1.5	Learning the $\lambda()$ Parameters	55
5.1.6	Relationship Pursuit	57
5.1.7	Analysis by Synthesis	58
5.1.8	Small Sample Set Generalization	59
5.2	Learning Results for Aerial Images	61
5.2.1	Data Collection	62
5.2.2	Object Representation	62
5.2.3	Relationship Functions	63
5.2.4	Deterministically Forming Parse Graphs	64
5.2.5	Learning the $\lambda()$ Parameters	67
5.2.6	Analysis by Synthesis	68
5.3	Conclusions on Learning Experiments	69
6	Inference With The C^4 Algorithm	71
6.1	Introduction	71

6.1.1	Motivations and Objective	71
6.1.2	Related Work in the Literature	74
6.1.3	Overview of the major concepts of C^4	76
6.2	Graphs, Coupling and Clustering	80
6.2.1	Adjacency and Candidacy Graphs	80
6.2.2	Positive and Negative Edges	82
6.2.3	The Necker Cube Example	83
6.2.4	Edge Probability for Clustering	86
6.3	C^4 algorithm on flat graphs	89
6.3.1	Outline of the algorithm	89
6.3.2	Calculating the Acceptance Probability	91
6.3.3	Special case: Potts model with +/- edges	93
6.4	Experiments on Flat Graphs	95
6.4.1	Checkerboard Ising Model	95
6.4.2	Checkerboard Potts Model with 7 Labels	97
6.4.3	Line Drawing Interpretation	99
6.4.4	Labeling Man-made Structures on CRFs	100
6.5	C^4 on Hierarchical Graphs	102
6.5.1	Condition for Graph Consistency	102
6.5.2	Formulation of Hierarchical C^4	104
6.5.3	Experiments on Hierarchical C^4	106
6.6	Conclusions on C^4	107

7	Experiments on Inference	109
7.1	Object Inference	109
7.2	Aerial Image Inference	113
7.2.1	Bottom-Up Detections	114
7.2.2	Top-Down Prediction of Missing Objects	120
7.2.3	Experimental Results	121
7.3	Conclusions on Inference Experiments	130
8	Discussion and Future Work	133
8.1	Summary of Major Contributions	135

LIST OF FIGURES

1.1	Examples of highly variant objects.	2
1.2	Examples of aerial images.	3
2.1	Four object classes and their corresponding manifolds in eigenspace. .	9
2.2	Example of a constellation model for the motorbike category.	12
2.3	A shock graph for silhouette matching.	14
2.4	An attributed grammar parse of an indoor scene.	15
2.5	An And-Or graph representation for clothing templates.	16
3.1	A visualization of a simple grammar for English.	20
3.2	A Markov Random Field on a grid.	24
3.3	Generation of an English sentence using pairwise Markov constraints.	24
3.4	A toy example of a contextual hierarchy.	26
3.5	Example of a contextual hierarchy for the clock class.	34
3.6	A contextual hierarchy for aerial images.	36
3.7	An example of parsing an aerial image with the contextual hierarchy. .	38
4.1	Visualization of learning the relationship parameters $\lambda^j(\beta)$	44
4.2	A visualization of the relationship pursuit procedure.	46
5.1	An example of the features computed for a single object.	52
5.2	Examples of corresponding bonding points for each terminal type. . .	53
5.3	Examples of relationships between object parts in our model.	54

5.4	Histograms of relationships during learning.	57
5.5	Examples of objects drawn from the model during learning.	58
5.6	Examples of objects sampled from our model for 24 classes.	59
5.7	Coverage results for 6 categories.	61
5.8	Demonstration of the model's generalizability.	62
5.9	Inter-object relationships in aerial images.	64
5.10	An example of deterministically forming parse graphs.	66
5.11	Histograms for four relations during learning.	67
5.12	Samples from our learned aerial image model.	69
6.1	Examples of problems with multiple solutions.	73
6.2	Swapping between the two interpretations of the Necker Cube.	77
6.3	Converting an adjacency graph to a candidacy graph.	81
6.4	The Necker cube adjacency graph.	84
6.5	A Necker cube candidacy graph not in a solution state.	88
6.6	The Potts model with negative edges.	93
6.7	The checkerboard Ising/Potts model solutions.	95
6.8	Comparison of C^4 energy and states on the Ising model.	97
6.9	Comparison of C^4 energy and states on the Potts model.	98
6.10	Experimental results for swapping state between interpretations.	100
6.11	Man-made structure detection results.	101
6.12	An attempt to solve the duck/rabbit illusion using flat C^4	103
6.13	Breaking the 'love triangle' in a candidacy graph.	104

6.14	An attempt to solve the duck/rabbit illusion using hierarchical C^4 . . .	105
6.15	Comparison of flat C^4 with hierarchical C^4 on the duck-rabbit example.	108
7.1	The edge strengths for positive and negative edges.	111
7.2	Hierarchical C^4 for detecting signal from noise.	112
7.3	Examples of good/medium/bad results for C^4	114
7.4	Plots of energy over time for C^4	115
7.5	Single object detections using our bottom-up detectors.	116
7.6	A conceptualization of inference with Compositional Boosting. . . .	118
7.7	An example of detecting roofs with Compositional Boosting. . . .	119
7.8	Top-down hallucinations of missing objects.	121
7.9	The bottom-up to top-down pipeline.	122
7.10	ROC curves for Compositional Boosting.	124
7.11	Close up views of C^4 improvement during top-down pruning. . . .	124
7.12	Close up views of C^4 improvement during top-down prediction. . . .	125
7.13	Precision-Recall curves for the inference algorithm.	127
7.14	An example of swapping alternative solutions using C^4	128
7.15	A comparison of results for models learned with different ϵ_1 s. . . .	129
7.16	Flat configurations of parsed images.	131
7.17	Flat configurations of parsed images.	132

LIST OF TABLES

5.1	Object relationship function definitions.	55
5.2	Relationship function definitions.	63
5.3	Category-dependent thresholds ϱ for deterministic grouping.	65
6.1	Inference results for man-made structure detection.	102
7.1	Detection improvement using our top-down model.	126

ACKNOWLEDGMENTS

First and foremost, I'd like to thank my advisor Song-Chun Zhu for his support, guidance, and friendship during my 5 year tenure at UCLA. I wouldn't have achieved half this much if I hadn't been working under his advisement. I would also like to thank my entire committee, Demetri Terzopoulos, Ying-Nian Wu, and Alan Yuille for their many insightful discussions and advice in completing this degree.

I would of course be remiss if I didn't acknowledge the incredible support of the other amazing students in the CIVS lab and their friendship during my time at UCLA - Brandon Rothrock, Siavosh Bahrami, Benjamin Yao, ZhangZhang Si, Kristy Wang, Leo Zhu, Song Feng Zheng, Iasonnos Kokinas, Zijian Xu, and the rest of the CIVS lab.

I also owe a huge debt of gratitude to the Statistics department at UCLA and all the wonderful mentors I had by my side in both the faculty and students - Mark Hansen, Jan de Leeuw, Rob Gould, Rick Paik Schoenberg, and everyone else who had time to talk shop with me.

Lastly, I couldn't have made it here without the constant support and encouragement of my family and friends. You guys deserve this more than I do.

Portions of this work were supported by an NSF grant IIS-0713652 and an ONR STTR grant N00014-07-M-0287. I thank the Lotus Hill Dataset for groundtruth annotation (Yao et al., 2007) for the testing images used in this dissertation.

VITA

1982	Born, Chatham, New Jersey, USA.
2003–2004	Undergraduate Research Assistant, Computer Graphics and User Interfaces Lab, Columbia University, New York, New York.
2003–2004	Undergraduate Research Assistant, Intelligent Agents Lab, Columbia University, New York, New York.
Winter 2003	Teaching Assistant, Computer Science Department, Columbia University. Programming Languages - C.
Winter 2003	Teaching Assistant, Computer Science Department, Columbia University. Programming Languages - C++.
Spring 2004	Teaching Assistant, Computer Science Department, Columbia University. Introduction to Programming in Java.
2004	B.S. (Computer Science, focus in Intelligent Systems), Columbia University.
Fall 2004	Teaching Assistant, Statistics Department, UCLA. Introduction to Statistics, 10A.
Summer 2005	Research Intern, Bell Laboratories, Murray Hill, New Jersey.
Fall 2005	M.S. (Statistics), UCLA, Los Angeles, California.
2005–Present	Graduate Research Assistant, Center for Image and Vision Sciences, UCLA.
Summer 2006	Research Intern, Google Inc., New York, New York.

Fall 2008	Teaching Assistant, Statistics Department, UCLA. Applied Statistics, 110A.
Winter 2008	Teaching Assistant, Statistics Department, UCLA. Applied Linear Regression, 101B.
2009–Present	Research and Development Scientist, UtopiaCompression Corporation, Los Angeles, CA.

PUBLICATIONS

J. Porway, K. Wang, S.C. Zhu, “A Hierarchical and Contextual Model for Aerial Image Parsing”, *International Journal of Computer Vision*, vol. 88, no. 2, pp. 254–283, 2010.

J. Porway, S.C. Zhu, “C4: Exploring Multiple Solutions in Graphical Models by Cluster Sampling”, *Pattern Analysis and Machine Intelligence*, submitted, 2010.

J. Porway, S.C. Zhu, “Automated Hierarchical Learning of Object Categories”, In preparation, 2010.

J. Porway, B. Yao, S.C. Zhu, “Learning Compositional Models for Object Categories From Small Sample Sets”, *Object Categorization: Computer and Human Vision Perspectives*, Cambridge Press, pp. 241–256, 2009.

L. Lin, T.F. Wu, **J. Porway**, Z. Xu “A Stochastic Graph Grammar for Compositional Object Representation and Recognition”, Pattern Recognition, vol. 42, no. 7, pp. 1297–1307, 2009.

J. Porway, K. Wang, B. Yao, S.C. Zhu, “A Hierarchical and Contextual Model for Aerial Image Understanding”, IEEE Conference on Computer Vision and Pattern Recognition”, pp. 1–8, 2008.

L. Lin, S. Peng, **J. Porway**, S.C. Zhu, Y. Wang, “An Empirical Study of Object Category Recognition: Sequential Testing with Generalized Samples”, Proceedings of the IEEE International Conference on Computer Vision, pp. 1 – 8, 2007.

S. Peng, L. Lin, **J. Porway**, N. Sang, S.C. Zhu, “Object Category Recognition Using Generative Template Boosting”, Energy Minimization Methods in Computer Vision and Pattern Recognition, vol. 4679, pp. 198–212, 2007.

ABSTRACT OF THE DISSERTATION

**A Hierarchical and Contextual Model for Learning and
Recognizing Highly Variant Visual Categories**

by

Jacob Matthew Porway

Doctor of Philosophy in Statistics

University of California, Los Angeles, 2010

Professor Song Chun Zhu, Chair

In this dissertation we present a hierarchical and contextual model for representing image patterns (manmade objects and aerial images) that are highly variant from instance to instance. These types of patterns are difficult to model because objects within the same class may have very different photometric and geometric properties and/or compositions of parts, e.g. teapots may have very different colors, shapes, and locations of their spouts and handles. We hypothesize that these varied visual patterns can be captured by using a novel representation that arranges common primitives of the patterns in a probabilistic hierarchy, thus compactly capturing possible compositional variations, and then enforces contextual constraints on the appearances of the parts, thus modeling the conditional photometric and geometric relationships of the object parts. We combine a Stochastic Context Free Grammar (SCFG), which captures the long-range compositional variations of a pattern, with a Markov Random Field (MRF), which captures the short-range constraints between neighboring pattern primitives, to create our model. We also present a minimax entropy framework for automatically learning which contextual constraints are most relevant for modeling a type of pattern and estimating their parameters. Finally, we present a novel Markov

Chain Monte Carlo (MCMC) algorithm called Clustering Cooperative and Competitive Constraints (C^4) for efficiently performing Bayesian inference with our model. C^4 is a method for minimizing energy functions defined on graphs that we will use to combine bottom-up and top-down information to find the best interpretation of an image. We show experiments on learning models of a number of manmade object categories and of aerial images and demonstrate that our algorithms automatically learn models that accurately capture the statistical nature of the patterns we are modeling. We also show that our model can be used for inference in new images, allowing it to identify objects in challenging scenarios.

CHAPTER 1

Introduction

This dissertation presents a hierarchical and contextual model for representing visual patterns with large variations in their photometric, geometric, and structural properties. We will present a grammar-based structure that embeds a Markov random field (MRF) in a stochastic context free grammar (SCFG) to efficiently capture these variations. Algorithms for performing learning and inference with this model are presented, followed by experiments showing the usefulness of this model for recognizing objects in new images.

1.1 Motivation

Many visual patterns in images exhibit huge intraclass variance, meaning that two instances of the same object may look drastically different. For example, two teapots may have entirely different colors, shapes, sizes, and configurations of their parts (e.g. one may have a high, curved handle as opposed to a low, straight handle). At times these objects may even look more similar to instances within other classes than to instances within their own class, e.g. bicycles designed to look like motorcycles.

Figure 1.1 shows some real world examples of common manmade objects that exhibit high intraclass variance. We can easily tell that the objects belong to the same class, despite the fact that have vastly different colors, shapes, sizes, and configurations of their parts. Figure 1.2 shows examples of urban aerial images, which are all easily



Figure 1.1: Examples of objects that exhibit high intra-class variance. All the bikes, clocks, and MP3 players look drastically different from one another, but we can recognize them as the same type of object

identified as overhead scenes of cities, yet that have vastly different numbers of buildings, cars, and roads, all of which in turn are different scales, colors, and shapes. In this dissertation we will explore the statistics of images within the same pattern class (e.g. clocks, aerial images) that allow us to recognize objects with different appearances and model these variations.

Despite the highly varied appearances of the objects in Figures 1.1 and 1.2 we can quickly identify consistencies between them. For example, though each clock in Figure 1.1 has a different shape, color, and arrangement of its hands, we observe that each clock consists of a frame, a set of hands, and some numbers. Similarly, the aerial



Figure 1.2: Examples of aerial images. Despite the high variance in color and composition, we can still recognize them as overhead views of cities.

images in Figure 1.2 all contain the same types of objects, namely trees, roads, cars, and buildings. In other words, each of these visual patterns, referred to henceforth as *image classes* or simply *classes*, are comprised of a consistent set of *primitives* or *parts*. In addition, we observe commonly obeyed relationships between the parts in each image. For example, clock hands are always centered within the frame, one hand is bigger than the others, and all the hands are the same color. Similarly in aerial images, cars appear on top of roads, buildings are bigger than cars, and cars don't park on top of trees. In other words, the parts of each class obey common *relationships* to one another in terms of their color, location, size, and overall relative appearance. We say that these parts obey *statistical constraints* on their appearances, or that they constrain one another. It is because of the commonly occurring parts in each class and their familiar relationships to one another that we can recognize variant instances as members of the same class.

1.2 Our Approach and Technical Contributions

To handle the huge variability present within image classes, we present a grammar-based **hierarchical and contextual model** for object recognition. This model arranges the primitives of each class in a probabilistic hierarchy and then represents the statistical constraints between them. The contextual hierarchy is not only an extremely expressive representation, but also serves as a great framework for combining bottom-up and top-down information during inference. In addition, we present a **minimax entropy learning framework**, previously applied in texture modeling, to learn the parameters of the model and a probabilistic clustering algorithm called **Clustering Competitive and Collaborative Constraints (C^4)** that is used to infer the presence of and relationships between object parts in images. These key contributions are outlined below:

A Hierarchical and Contextual Representation of Patterns

Our model for image classes is a grammar-based model that combines a stochastic context free grammar (SCFG) with a Markov Random Field (MRF) to capture both local and global context and to combine bottom-up information with top-down knowledge. SCFGs model highly variant patterns very succinctly by beginning with a single primitive and recursively generating new combinations of primitives according to weighted production rules until a full pattern is created. SCFGs are very compact representations of highly variant patterns as just a few production rules can be used to create a combinatorially huge number of instances. However, SCFGs lack the ability to capture local context. MRFs are often used to model local context, as they very efficiently model the probability over neighborhoods of variables. However, MRFs are defined only on local neighborhoods or related variables, and thus cannot capture higher-level, long-ranging context without becoming prohibitively complex. By fusing these two data structures, however, we create a model that can generate and recognize a huge number

of varied instances that are all guaranteed to be locally consistent. In our model, we represent the frequency of occurrence and type of object parts with a SCFG and model the spatial and appearance relationships between them using MRFs, thus creating a constrained grammar that can represent a huge number of instances for a single visual category.

A Minimax Entropy Learning Algorithm

Our contextual and hierarchical model learns statistical constraints on the appearances and relationships between different parts of the image class. Unlike most contextual models that require an explicit hand-defined set of relationships, our model uses a parsimonious minimax entropy framework to intelligently select the set of contextual relationships necessary for modeling the object class. The model begins with a huge set of relationships that could potentially exist between parts, but then iteratively selects only those relationships that help the model best match true statistics for that image class. By using this approach, one needs only provide some very basic definitions of contextual relationships (e.g. relative positions or relative orientation between parts), which will then be applied to every possible set of parts and automatically selected by the algorithm if they are deemed relevant. This learning algorithm both selects the most relevant contextual relationships for each object class and learns the parameters of the model in one concise process.

C^4 - A Probabilistic Clustering Algorithm for Inference

Our final contribution is an algorithm for inferring the presence of an object or objects within a new image. Given a set of candidate object parts that we detect in a new image, we must use the top-down information learned in our model to determine which truly arise because of an object and which are just false detections. The final piece of our research entails a probabilistic clustering algorithm called Clustering Cooperative and Competitive Constraints (C^4) that can quickly and efficiently identify which part

detections satisfy the learned constraints to comprise a full object. C^4 is an energy minimization algorithm defined on graphs that probabilistically activates cooperative, or positively associated, links between pairs of parts to form connected components (*ccps*) that represent sub-solutions to the problem. C^4 then identifies competitive, or negatively associated, connections between *ccps* to form *cccps* and then relabels entire *cccps* such that they satisfy learned constraints. The result is that, unlike greedy methods, C^4 can probabilistically jump out of local energy minima between different solutions (modes in the posterior) and can explore multiple solutions. Due to the fact that it probabilistically clusters potential object parts instead of analyzing each part individually, it can also move much more quickly through the solution space than traditional Metropolis-Hastings sampling methods. The end result is that the algorithm can quickly identify objects in images using local and global context to rule out false positives.

1.3 Overview of the Dissertation

In this dissertation we will describe the three major contributions listed above – a novel representation for highly variant object classes, a minimax entropy learning algorithm for automatically learning the model’s relationship constraints and their parameters, and an inference algorithm for finding objects in new images. The remainder of the dissertation is organized as follows:

Chapter 2 presents a survey of related work in object and aerial image modeling and highlights the shortcomings of current methods image class modeling and recognition.

Chapter 3 describes the representation of our hierarchical contextual model and presents its mathematical formulation. We will also discuss how to apply this model to images of man-made objects and aerial images.

Chapter 4 explains the minimax entropy learning algorithm for estimating the model parameters from a small set of labeled training images. We will examine how this process applies to both the object modeling and aerial image modeling.

Chapter 5 presents results on learning models of objects and aerial images using our method.

Chapter 6 provides the formulation and implementation of our inference algorithm, C^4 as well as experimental results showing C^4 applied to energy minimization problems on graphs.

Chapter 7 reports experimental results of recognizing objects and parsing aerial images using bottom-up detection methods in conjunction with C^4 for inference.

Chapter 8 concludes the dissertation with discussion of the results and an outline of future work.

CHAPTER 2

Related Work

The work in this dissertation focuses on algorithms for modeling and recognizing highly variant image classes. Previous solutions to this research problem can roughly be divided into *appearance-based methods*, *geometric-based methods*, and *compositional / grammar-based methods*. In this chapter we will describe related works from each of the three methods.

2.1 Appearance-Based Methods

The earliest attempts at object modeling fell primarily into the category of appearance-based methods. Appearance-based methods attempt to recognize objects and scenes using their photometric properties, i.e. by using either the pixels of the image directly or a set of features derived from the image. In focusing on the photometric qualities of the image, these approaches tend to disregard the geometric relationships within the image pattern, preferring instead just to use independent distinctive features in the image for classification.

In the early 90's Nayar (Nayar et al., 1996) learned a compact representation of objects by performing Principal Component Analysis (PCA) on images of each object under different poses and illuminations. PCA is a dimensionality reduction technique that computes an orthogonal basis of eigenvectors for a set of data such that the resulting eigenvectors are aligned in the directions of maximum variance. For a set of

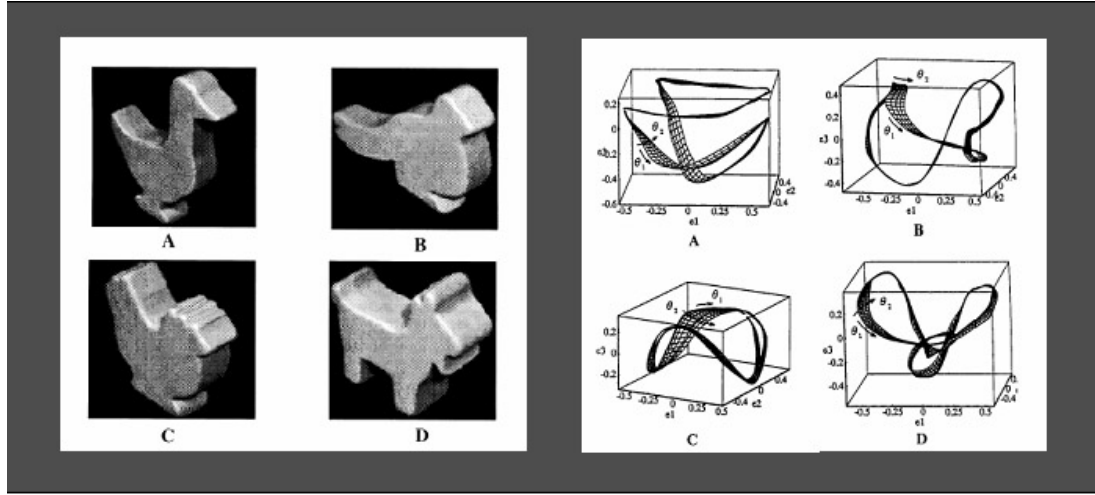


Figure 2.1: Four object classes and their corresponding manifolds in the eigenspace. The manifolds vary over pose and illumination.

training vectors X , the covariance matrix C can be computed as $C = X^T X$ and the eigenvectors computed by solving

$$Cv_i = X^T X v_i = \lambda_i v_i \quad (2.1)$$

where (λ_i, v_i) are the eigenvalues and eigenvectors. This technique achieves a high compression rate when the data points have high correlation with each other, which images of the same object do. Nayar mapped each image into its eigenspace, creating a manifold that varied with the pose and illumination of each object. Figure 2.1 shows a set of object classes and their resulting manifolds in eigenspace. Using this method, new images could be mapped to the eigenspace and their type, illumination, and pose determined based on where they existed in the space. Nayar's work (in turn inspired by Turk and Pentland's eigenfaces (Turk and Pentland, 1991)) would later give rise to kernel-PCA and other dimensionality-reduction methods for object recognition (Ali and Shah, 2005; Moghaddam, 1999).

Another appearance-based method that is quite popular today is the bag-of-words

classifier. The bag-of-words classifier encompasses a huge body of methods that represent image classes based on the occurrence of certain features (words) with no regard for the spatial correlation between these features. Common appearance-based features used for modeling image classes include SIFT (Lowe, 2004), SURF (Bay et al., 2008), or MSER (Forssen and Lowe, 2007) descriptors. The simplest classifiers simply use a Naive Bayes Classifier to determine whether an image contains a certain type of object or not. More sophisticated bag-of-words approaches use Latent Dirichlet Allocation (LDA) or probabilistic Latent Semantic Analysis (pLSA) to learn “topics” of groups of features (Blei and Jordan, 2003; Hoffman, 1999). Ullman used the presence of spatially fixed “patches” to identify the presence of objects (Ullman et al., 2001) and one of the most successful computer vision algorithms, AdaBoost (Freund and Schapire, 1997), learns a classifier based on weighted combinations of these low-level features.

Appearance-based methods, while simple to learn and, at times, quite successful for object categorization (Zhang et al., 2001), are simply too naive to capture the vast differences in appearances of objects within the same class. The lack of geometric and higher-level constraints between pixels and features means that appearance-based models will always be fooled by images that contain the right features but in the wrong order.

2.2 Geometric-Based Methods

Researchers recognized the shortcomings in purely appearance-based methods and began incorporating geometric constraints into the model. Geometric-based methods model objects as collections of parts that are related to one another by geometric constraints. Arguably one of the most famous among these models is the constellation model (Weber et al., 2000). The constellation model identifies common parts in an object category and models the relative position between them. In this way, the model

has much greater expressive power than appearance-based methods, as it models not just which parts are present, but where they should be in relation to one another. Fergus designed a representation that models the appearance, shape, and relative scale of a set of low-level features to one another (Fergus et al., 2003). Given a class model with learned parameters Θ , the likelihood of a new object appearing can be factorized as

$$p(X, S, A|\Theta) = \sum_{h \in H} p(X, S, A, h|\Theta) \quad (2.2)$$

$$= \sum_{h \in H} p(A|X, S, h, \Theta)p(X|S, h, \Theta)p(S|h, \Theta)p(h|\Theta), \quad (2.3)$$

where (X, S, A) are the position, shape, and appearance, respectively, and H is a set of hypothesis parts detected in a new image. The individual terms are modeled by independent Gaussians. Figure 2.2 visualizes the constellation model for the motorbike category. The plot shows the relative positions of the wheels, sides, and other features of the motorbike. The corresponding parts are indicated in the images below the plot. Variants and simplifications of the constellation model include the Star model (Fergus et al., 2005) K-Fans (Crandall et al., 2005), and pictorial structures (Felzenszwalb and Huttenlocher, 2005; Fischler and Elschlager, 1973).

While geometric methods are a large improvement over appearance-based methods, they still don't capture much of the variability present in different object classes. Geometric models capture the relative positions of a fixed number of parts, but lack flexibility in modeling the appearance of each part, the number of parts present, and the possible configurations of parts. For example, a clock may have a square or a pointed hand, which requires a more complicated likelihood model for the part than most part-based models use. Additionally, clocks can have two or three hands, depending on whether a second hand is present. The constellation model only understands a rigid set of parts, and thus has no flexibility to model both of these configurations. Lastly,

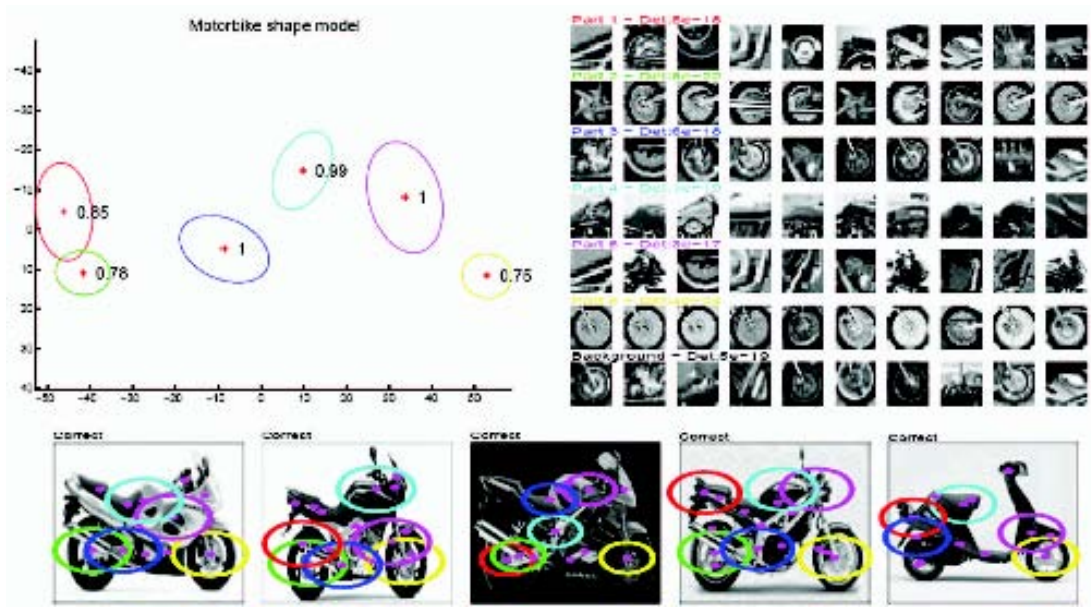


Figure 2.2: Example of the constellation model for a motorbike (Fergus et al., 2003). The plot shows the relative positions of the part distributions. The row of images below shows the presence of those parts in each image.

the relative positions of clock hands vary drastically over time. It will be very difficult to get a useful distribution of the hands' relative positions for the constellation model. Thus, while geometric methods are an improvement over appearance-based methods, their inability to model the various part configurations makes them infeasible for use with general object categories.

2.3 Grammar/Compositional-Based Methods

Recently grammar-based and compositional models have gained popularity in the machine learning and computer vision communities. These models seek not only to capture the relationships between objects or parts, but also arranges them hierarchically, allowing the number and configuration of parts to vary for each object. One can think

of grammars for images like grammars for a language - the grammar defines a dictionary of “words” (object parts) and “rules” (relationships between object parts) for how they need to be composed, thus generating an image of an object the way a grammar for language generates a sentence.

Early work by Fu(Fu, 1981) introduced attribute grammars to account for structural variance in images, but worked primarily on line drawings and shape contours, as opposed to natural images. Fu was one of the first to directly draw the comparison of image modeling to grammar models and to combine syntactic methods with statistical pattern recognition. These models are appealing because they explicitly model the relationships between fundamental parts of each class, or primitives, and gives an intuitive explanation for their generation. Dickinson (Dickinson et al., 1992; Keselman and Dickinson, 2001; Siddiqi et al., 1999) introduced shock graphs for modeling shapes in the 90’s, which arranged portions of medial axis skeletons into grammar trees (Figure 2.3). Up until this time however, there was little other work in grammar-based models for vision.

In the early 2000’s, grammar-based models made a resurgence for modeling visual categories. Han(Han and Zhu, 2005) used attributed graph grammars to describe rectilinear scenes as composed of groups of rectangles recursively composed by a set of arrangement rules. In Han’s work, the grammar’s only primitives were sets of rectangles and the spatial arrangements between them were hard-coded according to the decomposition rule used at each step. Figure 2.4 shows a visualization of Han’s indoor room modeling using rectangles and some of the decomposition rules that were hard-coded into the model. Chen (Chen et al., 2006) extended the attribute grammar work by introducing “composite templates” arranged in an And-Or graph to model clothes. Here, composite templates refer to deformable templates of different parts of clothes, e.g. the shirt sleeve, the collar, and the torso, and they were arranged into an And-Or

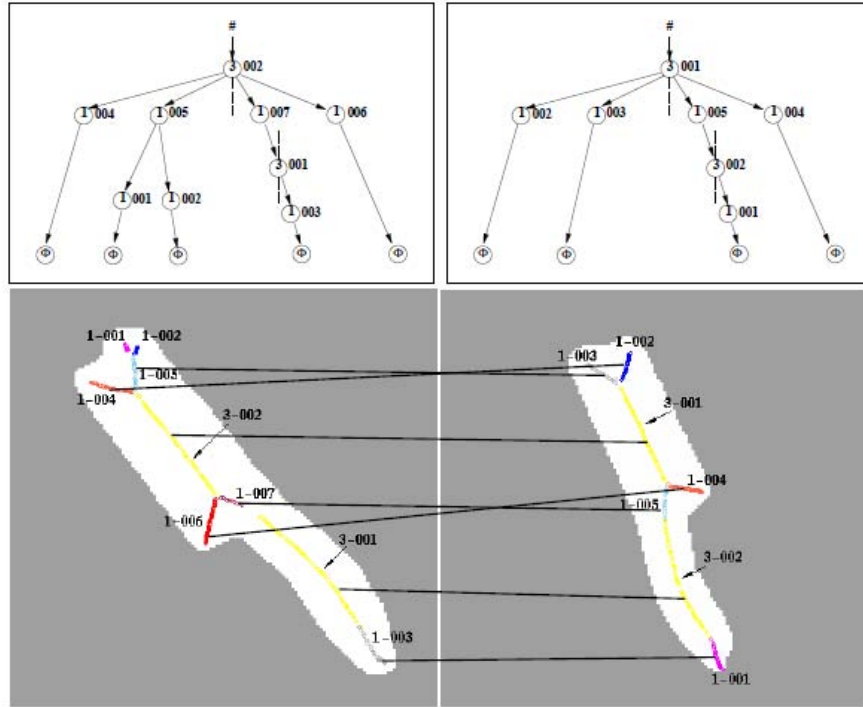


Figure 2.3: A shock graph for silhouette matching (Dickinson et al., 1992).

graph model (Pearl, 1984). Figure 2.5 shows an example of the And-Or graph model used for clothing. The hierarchy for different clothes types are hand-defined, as are the spatial and appearance relationships between them. Chen’s work is able to create and identify a huge variety of different clothing patterns, but requires a good deal of operator input that makes it infeasible to extend to arbitrary categories.

After 2005, the field of compositional models expanded dramatically. Winn’s work (Winn and Shotton, 2006) on Layout Consistent Random Fields learns a deformable configuration of object parts to represent objects within the same class that may have varied appearance. Todorovic created a graph-based model that models arbitrary regions in images and the relationship constraints between them (Todorovic and Ahuja, 2006). While this model is not a grammar-based model per se, it can automatically learn hierarchical compositions of object parts that are related to one another by

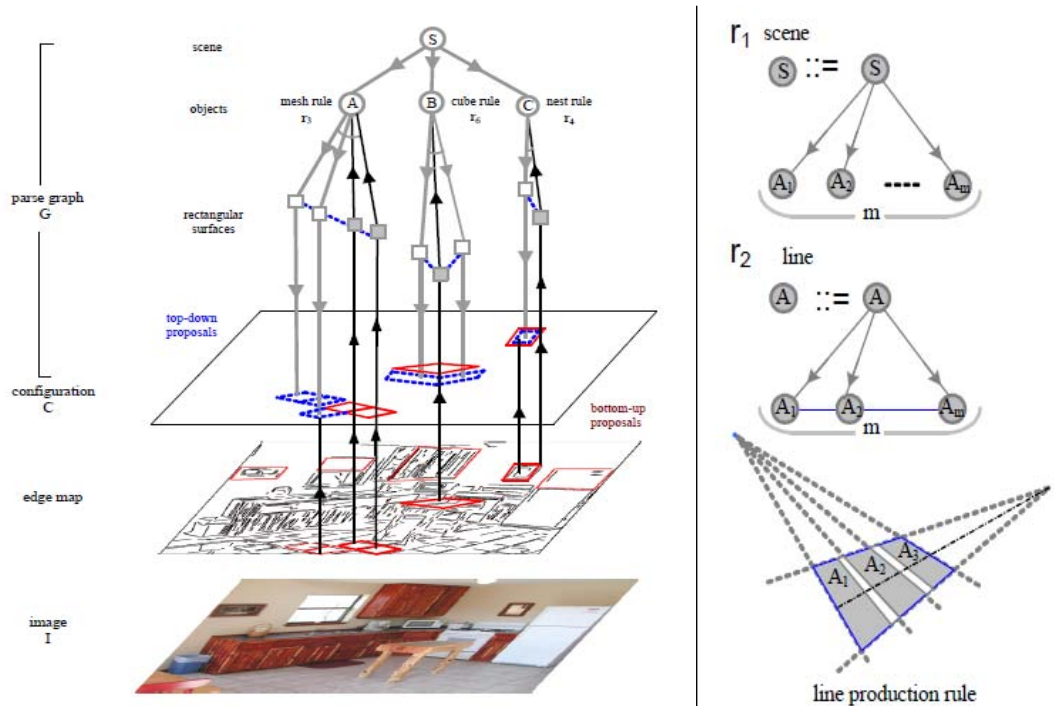
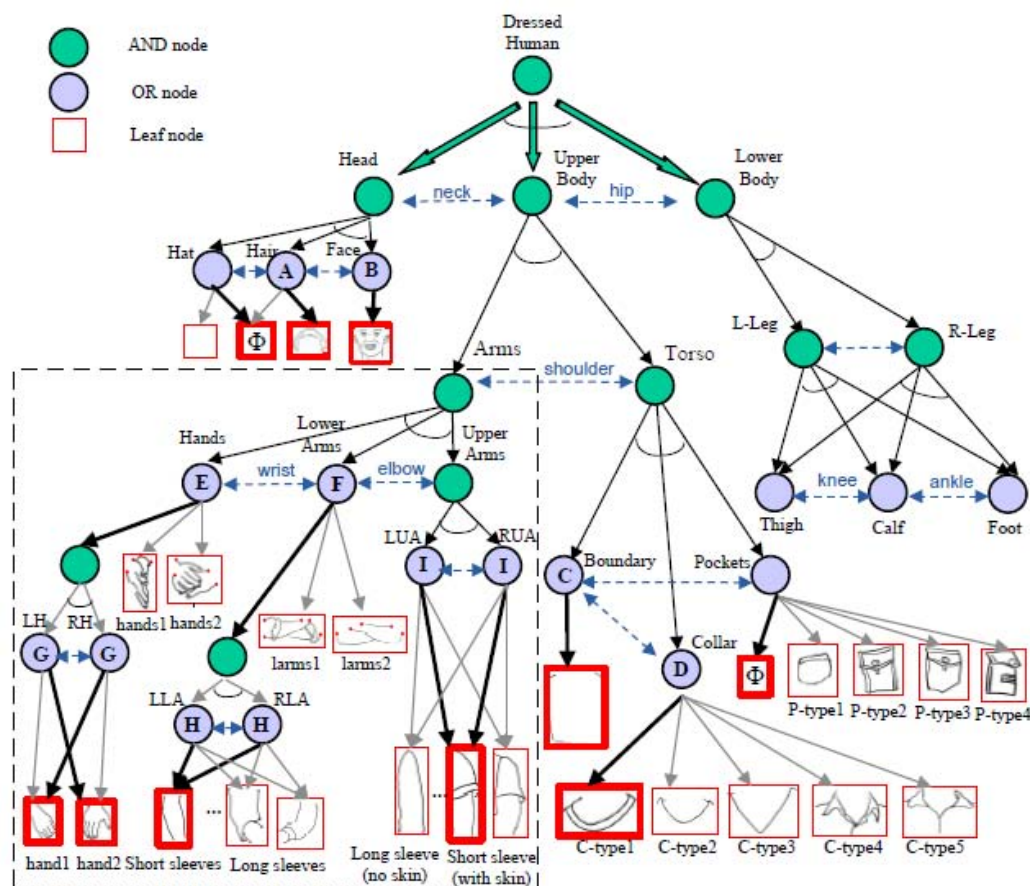


Figure 2.4: An attributed grammar parse of an indoor scene (Han and Zhu, 2005). The right panel shows different types of rectilinear production rules used in the grammar.

consistent appearance constraints. Sudderth used a hierarchical Dirichlet process to discover naturally occurring relationships between objects in scenes (Sudderth et al., 2005). Jin and Geman used a true grammar-based model to robustly identify license plates in images and read the text off of them, regardless of the variations in their appearances (Jin and Geman, 2006). Cao and Li created a spatial approach to common LDA in an attempt to model the relationships between parts identified by LDA (Cao and Li, 2007). Lastly, other work has created taxonomies of objects using reusable parts that captures hierarchy in terms of basic primitives and their relationships to one another (Torralba et al., 2004; Zhu et al., 2008; Sivic et al., 2005; Singhal et al., 2003; Li and Perona, 2005).

The works above are too vast and detailed to discuss individually in great depth.



However, we can note that our work takes inspiration from many of them in one form or another. The overall shortcoming in most of the methods above is that they lack something in terms of either representation, learning, or inference. Either their model does not consider both hierarchy and horizontal constraints, or there is not a simple way to learn the parameters of the model, or inference is made difficult by the complexity of the model. Our work seeks to address all three of these problems in a single framework.

2.4 Domain-specific Methods

As one of the domains we will apply our method to is aerial images, it behooves us to talk about the work that has been done in understanding objects in overhead imagery. As we will see, we can use our constrained grammar to model aerial images in terms of the objects present and their relationships to one another. Much work has been done on identifying single objects in aerial images, such as rooftops (Maloof et al., 2003; Vestri, 2001; Liu and Prinet, 2005), cars (Li et al., 2005; Zhao and Nevatia, 2001), or roads (Boichis et al., 2000). In these cases, however, context plays little role, as single objects are detected without taking the support of surrounding objects into consideration. These works use similar object detectors to those we will use, though they almost exclusively use one detector without considering the support from multiple detectors. These detectors include AdaBoost (Freund and Schapire, 1997; Viola and Jones, 2001), Bag of Words (Berg et al., 2007; Sivic et al., 2005) and TextonBoost (Shotton et al., 2006).

Some aerial imaging works incorporate context and/or multiple object category detection into the same framework. SIGMA (Matsuyama and Hang, 1990), a knowledge-based “expert system” for aerial images, was an attempt to model rule-based spatial relationships between objects. Unfortunately, as in much of the computer science based AI work of that time, relationships were often hardcoded and thus not generally extensible. On a smaller scale, Moissinac identified roads and city blocks in urban scenes using local context rules to decide how roads connect and how blocks should appear (Moissinac et al., 1994). Hinz used positional relations to determine the likely positions of roads in aerial images (Hinz and Baumgartner, 2000). A recent approach for parsing images of outdoor scenes by Berg (Berg et al., 2007) also seeks to model images as collections of regions that obey positional and relational constraints. As far as we know, however, these models require a good deal of hand-tuning and hardcoded

logic in order to encode the relevant constraints. SIGMA relied on experts to identify relationships of interest to model, Moissinac knew exactly the domain he was working with (handdrawn maps) and designed relationships accordingly, and Berg et al. used domain knowledge of the objects they wanted to identify to design contextual cues. Our model improves upon this shortcoming by employing a minimax entropy learning framework to automatically select significant relationships from a bank of potential relationships that can be designed to work on many domains of data without constant user input.

CHAPTER 3

The Hierarchical Contextual Model

The hierarchical contextual model that we present is composed of a stochastic context free grammar (SCFG) and a Markov random field (MRF). The SCFG captures the hierarchical nature of objects and images and allows for variability in the number and types of parts present. Given the objects from the tree-structure of the SCFG, the MRF components impose local constraints on the objects. For example, we may constrain car wheels to have the same size and color using an MRF. By allowing the SCFG to model the long-range relationships between object parts and the MRF to model the short-range local variability between object parts, the model can represent a huge range of potential appearances for an object in a principled way. We will describe the formulation of this model below.

3.1 Stochastic Context Free Grammars

Stochastic Context Free Grammars (SCFG) were extensively studied by Chomsky (Chomsky, 1956) in the 1950's in his work on modeling the structure of language. A Context Free Grammar (CFG) represents a language of patterns, defining the valid set of configurations for a dictionary of atomic tokens. More formally, a CFG G can be defined as a 4-tuple:

$$G = \langle V_N, V_T, R, S \rangle \quad (3.1)$$

where V_N is a set of non-terminal nodes, V_T is a set of terminal, or “leaf” nodes, R

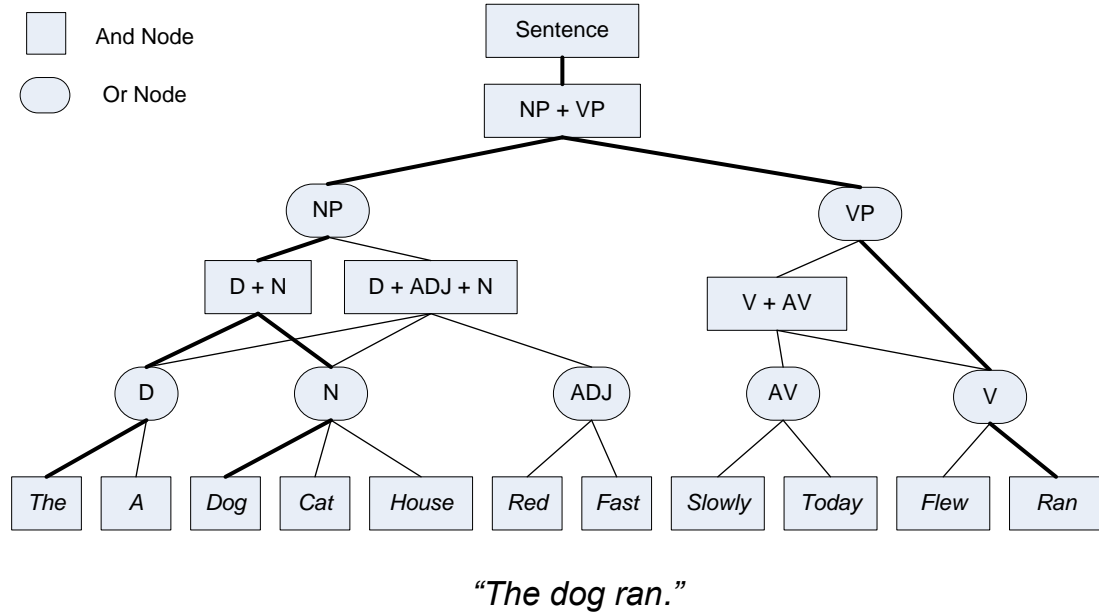


Figure 3.1: A visualization of a simple grammar for English. And nodes represent production rules and Or nodes represent sets of production rules that have the same symbol on the left. Dark thick lines show the parse tree for the sentence “The dog ran.”

is a set of production rules, and $S \in V_N$ is the root node. A production rule takes the form

$$R = V_N^i \rightarrow C \quad (3.2)$$

where C can be a conjunction of any number of non-terminal and/or terminal nodes. A CFG operates by beginning with the root node S , and decomposing it into a set of non-terminal and/or terminal nodes via the production rules in R . Each non-terminal node in the resulting decomposition is in turn transformed into a new set of non-terminal and/or terminal nodes via production rules until only terminal nodes remain. The end result is a “sentence” of all terminal nodes that were recursively generated via the production rules.

Figure 3.1 visualizes a simplistic CFG for the English language. Generating a

sentence entails starting at the root and applying production rules until a sentence is generated. For example, a sentence must consist of a noun phrase (NP) and a verb phrase (VP), so the first rule is

$$Sentence \rightarrow NP + VP \quad (3.3)$$

The non-terminal node NP could then be subject to one of two rules:

$$NP \rightarrow D + N \quad (3.4)$$

$$NP \rightarrow D + ADJ + N \quad (3.5)$$

$$(3.6)$$

This process of expansion continues until we end at the terminal leaf nodes.

Note that a grammar is nothing more than a set of nodes and a set of production rules that can be applied to those nodes. We are visualizing the full spectrum of potential applications of each production rule to each node in Figure 3.1. As such, let us introduce some terminology that will be very useful later. We define an *And node* as a non-terminal node that will decompose into one or more other non-terminal nodes. And nodes correspond to a single production rule that has one or more symbols on the right hand side. In our example, (NP + VP) is an And node as it must be rewritten as an NP AND a VP. We define an *Or node* as a non-terminal node that can decompose into a number of different sets of non-terminals. Or nodes represent the set of production rules that have the same symbol on the left side. For example, NP is an Or node because it can be rewritten as (D + N) OR (D + ADJ + N). We will use And nodes and Or nodes often in this dissertation while visualizing the possible decompositions of grammar models.

A CFG assumes that each non-terminal node V_N^i can be decomposed by uniformly selecting a production rule that has V_N^i on the left. To capture the true frequency with which terminals appear in the world, however, CFGs were extended to Stochastic

Context Free Grammars (SCFGs), which assign a probability $P = \{\rho_1, \rho_2, \dots, \rho_r\}$ to each production rule for it being chosen. This addition extends the 4-tuple from above to a 5-tuple,

$$G = \langle V_N, V_T, R, S, P \rangle, \quad (3.7)$$

which now includes the probability of selecting a certain production rule for a given non-terminal node.

In Figure 3.1, we can add a probability vector ρ_i at each Or node that models the probability with which it decomposes into each of its children. By recursively selecting these decompositions according to probability ρ_i , we walk through the tree and generate a realization from the grammar, or a sentence. We will refer to each realization from the grammar as a “parse tree”. The thick, dark lines in Figure 3.1 show the parse tree structure for the sentence “The dog ran”. The set of all possible sentences is known as the “language” of the grammar and is denoted $L(G) = \{w \in V_T^* : S \rightarrow^* w\}$, where w is a set of words and $S \rightarrow^* w$ indicates multiple productions starting at S and resulting in w .

The SCFG we have defined above is known as a random branching process in statistics. The probability of a parse tree pt consisting of a set of words $w \in V_T$ can be defined as

$$p(pt(w)) = \prod_{i=1}^{|w|} p(\rho_i). \quad (3.8)$$

Thus the probability of observing a given parse tree pt that creates a set of words w is simply the product of the production rule probabilities required to form that sentence. It is possible, of course, that there may be many parse trees that give rise to the same set of words w . Therefore, the probability of a given series of words w is simply the sum over all parse trees that produce the sequence w

$$p(w) = \sum_{pt(w)} p(pt(w)). \quad (3.9)$$

Thus, a grammar defines a probability distribution over a language of valid sentences, $L(G) = \{(w, p(w)) : S \rightarrow w, w \in V_T\}$.

Whether the grammar we have defined is over text, images, or another signal, we can see that the probability of creating a sentence treats the probability of forming each word independently. The probability of a word appearing depends only on its parent node, not on its neighboring words. Thus, SCFGs do not capture local context or any type of spatial interactions between the resulting words in the sentences.

3.2 Markov Random Fields

A Markov Random Field is a graph $\langle V, E \rangle$ on which a set of random variables X is defined that obey a group of Markov properties. The Markov properties are as follows:

1. **Pairwise Markov Property:** Any two non-adjacent variables are conditionally independent given all other variables:

$$X_u \perp X_v | X_{V/\{u,v\}} \text{ if } \{u, v\} \notin E \quad (3.10)$$

2. **Local Markov Property:** A variable is conditionally independent of all other variables given its neighbors:

$$X_v \perp X_{V/cl(v)} | X_{\delta\Lambda(v)} \quad (3.11)$$

where $\delta\Lambda(v)$ is the set of neighbors of v and $cl(v) = \{v\} \cup \delta\Lambda(v)$ is the closed neighborhood of v .

The Markov properties are a mathematically precise way of describing local context. The second property is particularly useful and intuitive, and states that the probability of a given variable's state is dependent only upon its local neighborhood. Thus, MRFs are perfect representations of models that account for local context.

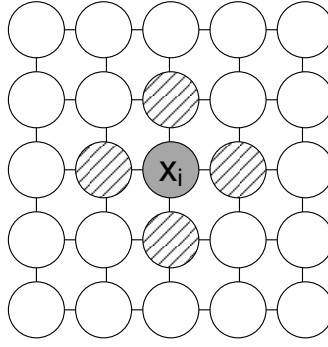


Figure 3.2: A Markov Random Field on a grid. The probability of x_i 's value is independent of the rest of the grid given its neighbors (shown in gray hatch).

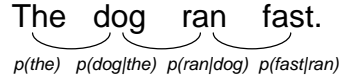


Figure 3.3: Generation of an English sentence using pairwise Markov constraints.

Figure 3.2 shows an example of an MRF grid. The probability $p(x_i|X) = p(x_i|\Lambda_{x_i})$, meaning that the probability of x_i is conditionally independent of the rest of the graph X given its neighborhood Λ_{x_i} .

By the Markov property, we can define the probability of a sentence of words w by simply modeling the probability of each word given its previous word,

$$p(w) = p(w_0) \prod_{i=1}^{N-1} p(w_i|w_{i-1}). \quad (3.12)$$

In this way, MRFs incorporate local spatial constraints into the model. Figure 3.3 shows a pairwise generation of a sentence of English words. Where the SCFG modeled the structure of a sentence as a composition of different sub-parts but didn't account for spatial interactions between neighboring words, the MRF model accounts explicitly for the probability of words given their neighbors.

We can also define the probability for an MRF over a set of variables X as a product of their *clique potentials* $\phi_C(x_C)$, where a clique $x_C \in X$ is merely a subset

of the graph G and we know that we can factorize G into a set of cliques. The clique potentials $\phi_C()$ are real-valued functions over the variables in each clique, and the full probability can be defined as

$$p(X) = \frac{1}{Z} \prod_{C \in G} \phi_C(x_C), \quad (3.13)$$

where Z is a normalization constant over all possible configurations of the variables \mathcal{X} defined as

$$Z = \sum_{X \in \mathcal{X}} \prod_{C \in G} \phi_C(x_C). \quad (3.14)$$

This formulation is useful for graphs that are not as linear as our sentence example and are more easily defined in terms of functions over sets of variables than on the single variables themselves.

While the MRF captures local probabilities and configurations, it cannot ensure global consistency, e.g. that the words more than two neighbors away are consistent with each other.

3.3 Creating a Contextual Hierarchy

We have shown that SCFGs are very powerful for representing the variability of all the different instances in a language, or class of patterns, however they lack local consistency. We have also shown that MRFs capture local consistency, but have trouble modeling global context and variability. Additionally, the examples we've given so far have made the connection to grammars for language, but we have yet to discuss the extension of these models to images and visual patterns. Sentences in English are formed from well-known tokens (words) in a linear format (a sentence), but images have no such agreed-upon atomic unit and exhibit far more complex correlations than simply linear order. We will now show how to form a constrained hierarchy by combining SCFGs and MRFs and describe their application to the visual domain.

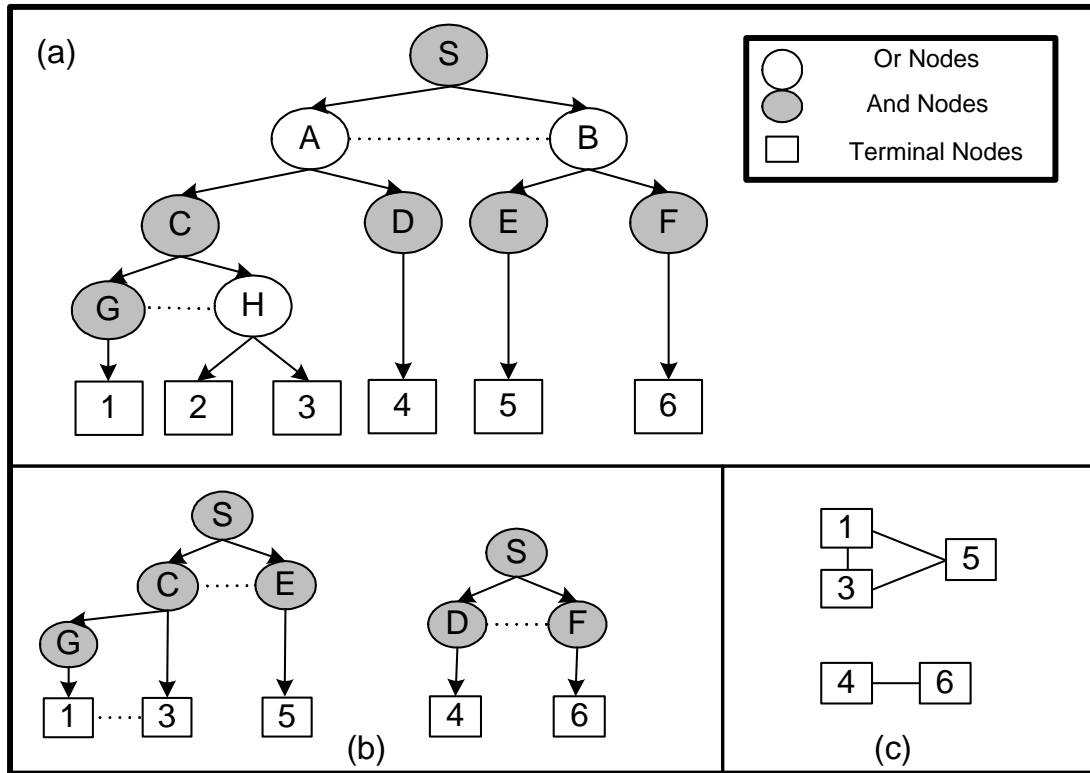


Figure 3.4: A toy example of a contextual hierarchy. (a) The full model for this language. Nodes are labeled as And or Or nodes. Dotted lines indicate contextual relationships. (b) Two parse graphs generated from the model. (c) Flattened parse graphs are configurations or realizations from the model, like sentences.

Figure 3.4(a) shows an example of a contextual hierarchy. An object is created by starting at the root of the graph and expanding nodes until only terminals remain, as in a SCFG. Just as in an SCFG node expansions consist of “And” nodes, where one node expands into multiple nodes and “Or” nodes, which can only choose one of their child nodes to expand into. For example, node *S* is an And node, and expands into nodes *A* and *B*, which in turn are Or nodes and will only decompose into one child each. Figure 3.4(a) is a visualization of the following grammar

$$\begin{array}{lll}
S \rightarrow AB & C \rightarrow GH & F \rightarrow 6 \\
A \rightarrow C \mid D & D \rightarrow 4 & G \rightarrow 1 \\
B \rightarrow E \mid F & E \rightarrow 5 & H \rightarrow 2 \mid 3
\end{array}$$

Unlike a traditional SCFG, however, we incorporate relational constraints between pairs or cliques of nodes in the tree. The horizontal line between A and B represents a relational constraint. These constraints do not influence the node expansion, but act *a posteriori* on the selected nodes to constrain their features, e.g. appearance. The constraints are inherited by any children of constrained nodes as well.

The contextual hierarchy is formulated similarly to the 5-tuple for the SCFG,

$$\mathcal{G} = \langle V, R, P \rangle, \quad (3.15)$$

where $V = S \cup V_N \cup V_T$ is our full set of non-terminal and terminal nodes, R are now statistical relationships between the nodes, instead of production rules, and P is our probability model over the graph.

V : Terminal and non-terminal nodes: We've compressed the root node, terminal nodes, and non-terminal nodes into a single set V for convenience,

$$V = S \cup V_N \cup V_T. \quad (3.16)$$

In the language example, terminals were words and non-terminals were parts of speech. For visual patterns, terminals will be image structures, such as part boundaries or low-level descriptors. The non-terminal nodes are simply compositions of lower-level image structures.

Each non-terminal node $V_N = \{V^{And} \cup V^{Or}\}$ is either an And node or an Or node. We define a variable $\omega(v_i)$ on Or nodes $v_i \in V^{Or}$ that takes an integer value indexing which of its $N(\omega(v_i))$ children it can decompose into,

$$\omega(v_i) = k; k = \{1, 2, \dots, N(\omega(v_i))\}. \quad (3.17)$$

(Note: we will often use $N()$ throughout this dissertation to represent the cardinality of certain quantities.) $\omega(v_i)$ acts as an index into the production rule chosen at each node. Thus, the probability $p(\omega(v_i) = k)$ is the probability that node v_i will decompose into its k^{th} child.

Regardless of what type of image structure we define for the terminal and non-terminal nodes, each node $v_i \in V$ will have a set of attributes $\phi(v_i)$. In general, these attributes will at least include the part's position, scale, and orientation,

$$\phi(v_i) = \{X_i, \sigma_i, \theta_i\} . \quad (3.18)$$

In Sections 5.1.2 and 5.2.2 we present our implementation of these attributes for our experiments.

R : Statistical Relationships: Instead of production rules, $R = \{r_1, r_2, \dots, r_{N(R)}\}$ defines the set of relationships that exist between nodes at the same level of the hierarchy. These relationships are constraints on the attributes of sets of nodes, for example how far apart they are, whether they're the same color or not, or how different their sizes are.

A relationship r_i consists of a set of k nodes $V_i \in V$ that it acts on, a univariate function $f()$ over their attributes that acts as a statistical constraint, and a model of the responses of $f()$, p :

$$r_i = \{V_i, f_i(\phi(V_k)), p_i\} . \quad (3.19)$$

For example, the relative position between cars and buildings could be expressed as

$$f(\phi(Cars), \phi(Buildings)) = X_{Cars} - X_{Buildings} . \quad (3.20)$$

If we believe relative position between cars and buildings is normally distributed with mean 5 and standard deviation 1, then the whole relationship is packaged as

$$r_i = \{(Cars, Buildings), f_i = X_{Cars} - X_{Buildings}, f_i() \sim p_i = \mathcal{N}(5, 1)\} . \quad (3.21)$$

We will discuss the relationship functions $f()$ and their distributions p_i in our implementation in Sections 5.1.5 and 5.2.5. At this point it is enough to know that each relationship represents the distribution of a function response over a set of nodes. These distributions act as our statistical constraints.

pg : Parse Graphs: A realization from an SCFG was called a parse tree and comprised a subset of a tree. For our contextual hierarchy we change our definition slightly and define a “parse graph”, which is a subset of the contextual hierarchy. A parse graph corresponds to a single realization from our model (e.g. a single clock), but now includes the horizontal constraints between nodes, hence the term parse “graph”. Figure 3.4(b) shows an example of a parse graph drawn from our contextual hierarchy for the toy example. The Or nodes are determined during a walk of the graph, fixing the parts, so only And nodes remain. The And nodes have been constrained by the corresponding relationships learned in the contextual hierarchy. By flattening the parse graph hierarchy, we get a single object, or *configuration*, like a sentence in a language, consisting of the parts we have stochastically selected and their obeyed statistical constraints. This is equivalent to a sentence in natural language, and Figure 3.4(c) shows examples of configurations from our example.

Let us define some terminology on parse graphs, similar to that of \mathcal{G} :

1. $V_{pg} \subseteq V$: The nodes present in pg , which are a subset of the nodes possible in \mathcal{G} .
2. $\Omega_{pg} = \{\omega(v_i); v_i \in V_{pg}^{Or}\}$: The values of the production rules selected to form pg . For example, if the node A from Figure 3.4(a) decomposed into D , its second child, then $\omega(A) = 2$.
3. $R_{pg} \subseteq R$: The constraints, or edges, between nodes in V_{pg} . These edges are inherited from the relationships R present in \mathcal{G} , so if the model has learned that

clock hands should be in the center of the frame, the clock hands we created will be constrained to appear in the center of the frame we created.

P : Probability Model: P is our probability model, including the probability that non-terminal nodes decompose into a certain number of child nodes, as well as the probability encoded in our statistical constraints, i.e. the probability that two object parts appear under certain relationships to one another. We will define the probability model for the contextual hierarchy as a probability of observing a single parse graph.

3.4 Mathematical Formulation for the Contextual Hierarchy

We observe a set of N^{obs} images $I^{obs} = \{I_i^{obs} : i = 1, 2, \dots, N^{obs}\}$ drawn from the class that we are trying to model. These images have corresponding parse graph representations $PG^{obs} = \{pg_i^{obs} : i = 1, 2, \dots, N^{obs}\}$ associated with them. The parse graphs pg follows some true, unknown distribution, $f(pg)$, determined by nature. Our goal is to design a model $p(pg)$ that approximates $f(pg)$ as closely as possible. One way to do this is to match the marginal statistics of $p(pg)$ to the marginal statistics of $f(pg)$. If for some statistic $\phi(pg)$, $E_p[\phi(pg)] = E_f[\phi(pg)]$, then our model $p(pg)$ recreates the observed statistics of $\phi(PG)$ and is similar along that dimension to $f(pg)$. If this is true for many $\phi()$, then $p(pg)$ is a functional approximation to $f(pg)$.

The observed statistics of $f(pg)$ that we would like our model $p(pg)$ to recreate are

1. The distribution of $\omega(v_i)$, the number of children each node $v_i \in V^{Or}(pg)$ decomposes into.
2. The distribution of responses of $f_i()$ for each statistical relation $r_i \in R$ in \mathcal{G} .

We can model the distributions of these variables as histograms. Our motivation for using histograms is two-fold: They are piecewise constant approximations to the con-

tinuous distributions we are modeling in the limit and they don't require knowing the form of the distributions in advance. We can thus describe trimodal distributions as easily as unimodal distributions by using histograms. We will discuss the implementation of our histogram representation in Sections 5.1.3 and 5.2.3.

We pool our Or node switch variable values $\omega(v_i)$ and relationship values $f_i()$ observed in PG^{obs} into histograms,

$$H_i^{(\alpha)}(pg, z) = \frac{\sum_{k=1}^{N^{obs}} \#(\omega(v_i) = z)}{\sum_{k=1}^{N^{obs}} \sum_{j \in N(\omega(v_i))} \#(\omega(v_i) = j)}, i = 1, 2, \dots, N(V^{Or}(pg)) \quad (3.22)$$

$$H_j^{(\beta)}(pg, z) = \sum_{k=1}^{N^{obs}} \frac{\sum_{V_j \subseteq V_{pg_k}} \#(f_j() = z)}{\sum_{V_j \subseteq V_{pg(i)}} \#(f_j())}, j = 1, 2, \dots, N(R) \quad (3.23)$$

where $\#$ is a counting function representing the number of times that something occurs, and $\#(f_i())$ is the number of times $f_i()$ takes any value. Note that we use the superscript (α) to refer to histograms of switch variables and (β) to refer to histograms of relationship function responses. Each bin z in $H_i^{(\alpha)}(pg)$ is then the number of times that node v_i decomposes into child node z divided by the number of times we observe v_i decomposing into any of its $N(\omega(v_i))$ children. Each bin z in $H_i^{(\beta)}(pg)$ is the number of times that relationship function $f_i()$ returns z divided by the number of times $f_i()$ returns anything. These histograms are simply the values we observe for each parse graph pooled over PG^{obs} .

We want to design our model such that the expectation of these distributions with respect to our model $p(pg)$ match the expectation of these distributions with respect to the observed distribution $f(pg)$.

$$E_p[H_i^{(\alpha)}] = E_f[H_i^{(\alpha)}]; i = 1, \dots, N(V^{Or}) \quad (3.24)$$

$$E_p[H_j^{(\beta)}] = E_f[H_j^{(\beta)}]; j = 1, \dots, N(R) \quad (3.25)$$

Note that we will not have access to $E_f[H_i^{(\alpha)}]$ or $E_f[H_j^{(\beta)}]$ directly, as f is unknown, but we can estimate them using the sample means of the histograms of the

observed data,

$$E_f[H_i^{(\alpha)}] \approx \frac{1}{N^{obs}} \sum_{k=1}^{N^{obs}} H_i^{(\alpha)}(pg_k^{obs}) \quad (3.26)$$

$$E_f[H_j^{(\beta)}] \approx \frac{1}{N^{obs}} \sum_{k=1}^{N^{obs}} H_j^{(\beta)}(pg_k^{obs}) . \quad (3.27)$$

We will intend this approximation whenever we refer to expectations with respect to f for the remainder of the dissertation.

Of all of the distributions that match the Or node and relationship statistics of the observed data $\Omega_p = \{p(pg) : E_p[H_i^{(\alpha)}] = E_f[H_i^{(\alpha)}], E_p[H_j^{(\beta)}] = E_f[H_j^{(\beta)}]\}$, we want to select the one that is least biased or most unprejudiced among any other dimensions. We can phrase our search for the optimal distribution $p(pg)^*$ as the following constrained optimization problem:

$$p(pg)^* = \arg \max \left\{ - \sum p(pg) \log p(pg) \right\} \quad (3.28)$$

subject to

$$E_p[H_i^{(\alpha)}] = E_f[H_i^{(\alpha)}], \quad i = 1, 2, \dots, N(V^{Or} \in \mathcal{G}) \quad (3.29)$$

$$E_p[H_j^{(\beta)}] = E_f[H_j^{(\beta)}], \quad j = 1, 2, \dots, N(R \in \mathcal{G}) \quad (3.30)$$

Via maximum entropy, the probability model that satisfies these constraints and reproduces the observed statistics is the familiar Gibbs model

$$p(pg; \Theta, R) = \frac{1}{Z[\Theta]} \exp^{-\xi(pg)} \quad (3.31)$$

$$\xi(pg) = \sum_{i=1}^{N(V^{Or})(pg)} \langle \lambda_i^{(\alpha)}, H_i^{(\alpha)}(pg) \rangle + \sum_{j=1}^{N(R)(pg)} \langle \lambda_j^{(\beta)}, H_j^{(\beta)}(pg) \rangle \quad (3.32)$$

$$Z[\Theta] = \sum_{pg \in L(\mathcal{G})} \exp^{-\xi(pg)} , \quad (3.33)$$

where $\Theta = \{\lambda^{(\alpha)}, \lambda^{(\beta)}\}$, R is the set of contextual relationships, and $Z[\Theta]$ is a normalization constant. The first term in $\xi(pg)$ is the energy of the Or node decompositions and the second is the energy of the relationship constraints. If we have an unlikely set of parts in our image (e.g. a clock with no hands or an oddly shaped clock frame), then the first term will have high energy and the interpretation will have low probability. If we have objects that do not obey the statistical constraints we learned during training (e.g. a clock with hands that aren't centered), then the second term will have high energy and the interpretation will have low probability.

The Lagrange multipliers $\{\lambda^{(\alpha)}, \lambda^{(\beta)}\}$ are vectors of the same dimension as $H^{(\alpha)}$ or $H^{(\beta)}$, respectively, and $\langle \dots \rangle$ indicates an inner product. For example, if relationship $r^{(\beta)}$'s function f_i evaluates to z on parse graph pg , then the energy from that relationship is $\lambda^{(\beta)}(z) * H^{(\beta)}(pg, z)$. The λ 's are the natural parameter set of the model and serve to weight histogram bins so that dependent relationship interactions are weighted correctly. These λ 's will be learned in the following chapter.

3.5 Application to Object Modeling

Figure 3.5(a) shows a contextual hierarchy for the clock object class. The terminal nodes consist of low-level object parts that are combined into higher-level compositions at the non-terminal nodes. We have chosen to use conceptual object parts, such as clock hands or frames, as the leaves of the tree, though one could extend the model further by decomposing these parts into their constituent visual primitives, such as edges or blobs. The relationships between parts are statistical constraints on their physical appearance, primarily focused on modeling relative position, relative scale, and relative orientation. Figure 3.5(b) shows two parse graphs from the clock class. The dark arrows in Figure 3.5(a) indicate the Or node productions used to create the top parse.

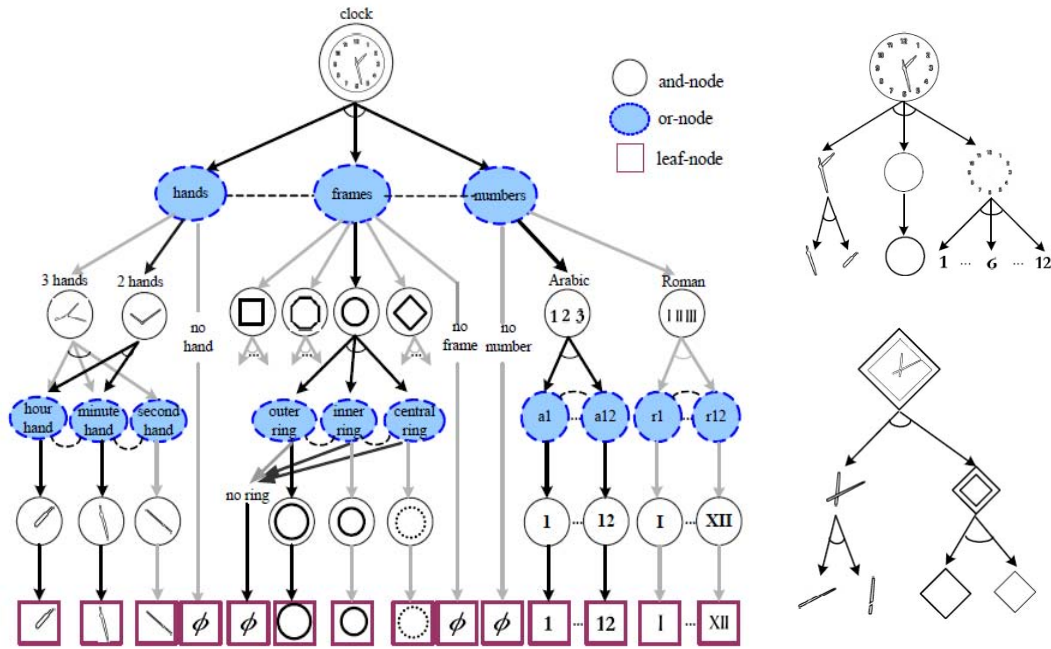


Figure 3.5: (a) A contextual hierarchy for the clock class. The root level object (an And node) decomposes into a set of constituent parts that can each take varied appearances (Or nodes). The dashed horizontal lines represent statistical constraints between the parts. (b) Two valid clock parse graphs drawn from the model.

By flattening the hierarchy, we get the familiar images of clocks at the roots of each parse graph.

The hierarchy that we construct for each object class is currently defined by hand. This is not terribly time consuming, as many of the object classes only consist of a small number of different parts. We can therefore define the high-level parts we are interested in (e.g. clock hands and clock frames) and arrange them hierarchically. We are considering techniques to help automate this process and learn the hierarchy automatically.

3.6 Application to Aerial Image Modeling

Unlike images of objects, each aerial image does not have a set number of parts that simply need to be reconfigured. Aerial images instead consist of varying numbers of common objects, such as cars, trees, and roads. To model aerial images, we define a core set of object types and let the non-terminal nodes in the hierarchy indicate single objects or groups of objects. Figure 3.6 shows this modified hierarchy. The model consists of nodes divided into a root scene node, group nodes, and object nodes.

Group nodes are collections of the same type of object, such as blocks of roofs or lines of cars, while object nodes are the single objects within each group. Below this level is the object representation level, which may be hierarchical in and of itself, as in the case of roofs and roads in our example, or may terminate at a one-layer representation for the object. The top 3-levels are representation agnostic, however, so we will put off a discussion of object detection and representation until Section 7.2.1. The thick arrow edges between the scene node and group nodes and between the group nodes and the object nodes indicate that a varying number of each group node may be present, and the number of object nodes they are comprised of can vary as well. The number of objects within each group and number of groups in the scene are our implicit Or nodes in the model, while the groups and objects themselves are the And nodes.

The aerial image hierarchy is similar to our object grammar, where the scene node decomposes into a variable number of object groups, which in turn decompose into a variable number of objects. This captures the loose, variable nature of aerial images with just a few compact rules. If we were to write these expansions in a grammar

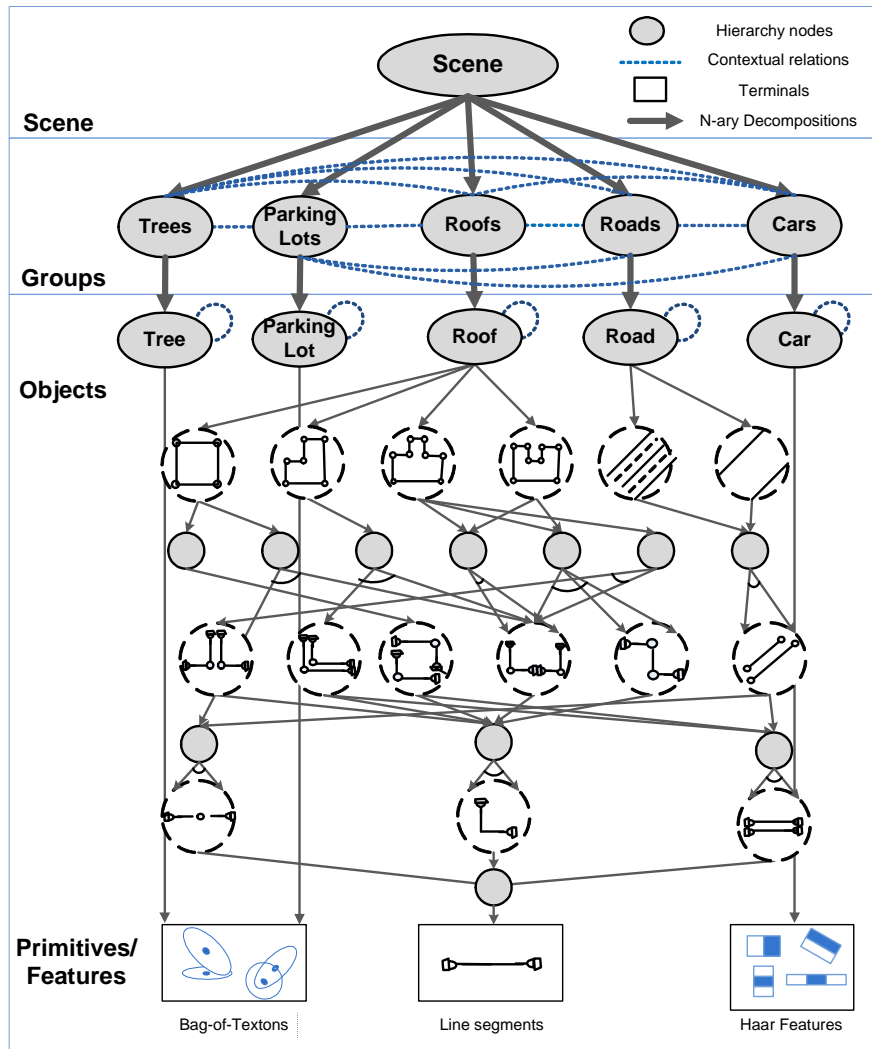


Figure 3.6: A contextual hierarchy for aerial images. Instead of having a small set of specific parts, common objects (e.g. cars, roads, trees) are grouped hierarchically. Statistical constraints exist between objects and groups. The individual objects can be decomposed further, like roofs, into more detailed compositions.

format, we would write

$$Scene \rightarrow (Roads^*) \cup (Roofs^*) \cup (Trees^*) \cup (ParkingLots^*) \cup (Cars^*) \quad (3.34)$$

$$Roads \rightarrow Road^*$$

$$Roofs \rightarrow Roof^*$$

$$Trees \rightarrow Trees^*$$

$$Parking\ Lots \rightarrow Parking\ Lot^*$$

$$Cars \rightarrow Car^* .$$

Here we're using “*” in the regular expression sense, meaning 0 or more of an object. One could rewrite the “*” operator by enumerating all cases, as in

$$Roads \rightarrow \emptyset | Road | (Road)(Road) | (Road)(Road)(Road) | \dots \quad (3.35)$$

Figure 3.7 shows an example of parsing an aerial image using this model. The original image (a) is decomposed into individual objects (b) that are then grouped together via statistical relationships into a parse graph of the scene (c). The dashed horizontal lines show the relationship constraints between groups and objects. Figure 3.7(d) shows which objects would be related by certain relationships, such as alignment and relative position.

In object modeling, our Or nodes indicated which type of object part a node would decompose to. In aerial image modeling, our top-level Or nodes describe the number of groups each scene decomposes to and, in turn, the number of objects each group decomposes into. Beyond the object level one can include additional And and Or nodes to describe the variation in each object's appearance as we did for object modeling. We will discuss the implementation of the object representations in Section 5.2.2 and relationships in Section 5.2.3.

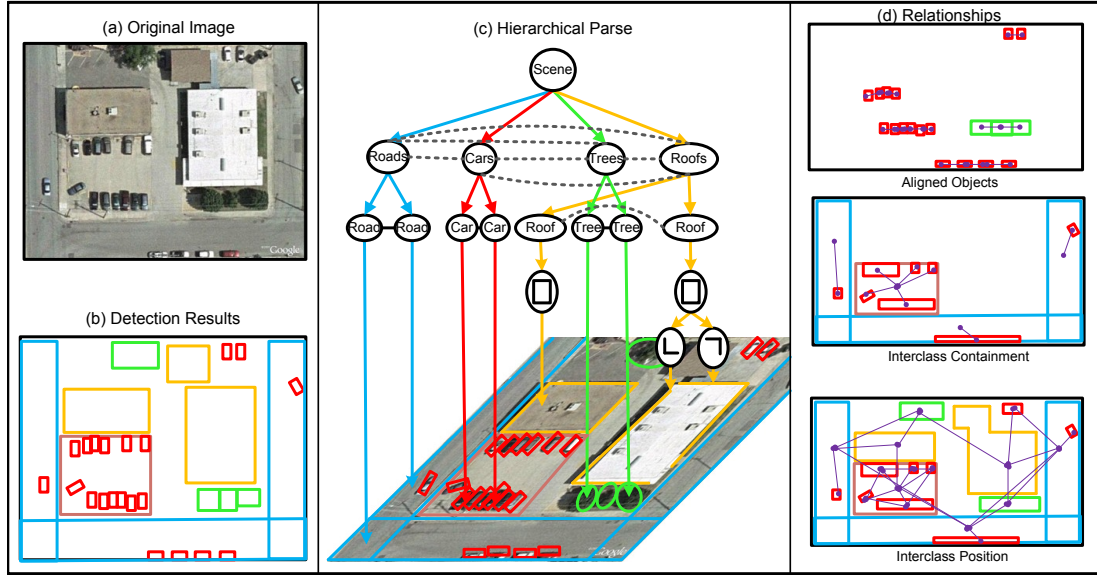


Figure 3.7: An example of parsing an aerial image with the contextual hierarchy. (a) Original image (b) Objects are detected in the image (c) Objects are grouped via statistical constraints according to the model to form a parse of the scene. Dashed horizontal lines indicate constraints between groups and objects. (d) Objects related by certain types of relationships.

CHAPTER 4

Learning via Minimax Entropy

In this chapter we present an efficient minimax entropy algorithm for both learning which relationships are relevant for modeling each category and estimating the relationship parameters.

4.1 Maximum Likelihood Estimation

We begin with a set of N^{obs} images $I^{obs} = \{I_i^{obs} : i = 1, 2, \dots, N^{obs}\}$ and their corresponding parse graphs $PG^{obs} = \{pg_i^{obs} : i = 1, 2, \dots, N^{obs}\}$. The parse graphs PG^{obs} follow the real-world, unknown target distribution, $f(pg)$, by definition,

$$pg_i^{obs} \sim f(pg) . \quad (4.1)$$

Matching our distribution $p(pg)$ to $f(pg)$ is equivalent to finding the values for Θ that minimize the KL divergence between the two distributions

$$\begin{aligned} \Theta^* &= \operatorname{argmin}_{\Theta} KL(f(pg) || p(pg; \Theta, R)) \\ &= \operatorname{argmin}_{\Theta} \sum_{pg} f(pg) \log \frac{f(pg)}{p(pg; \Theta, R)} , \end{aligned} \quad (4.2)$$

which is equivalent to finding the maximum likelihood estimates for Θ and a set of relationships R constraining the model. Letting $\mathcal{L}(\Theta)$ be the log-likelihood function

for our parameters,

$$\mathcal{L}(\Theta) = \sum_{k=1}^{N^{obs}} \log p(pg_k; \Theta, R) \quad (4.3)$$

$$= \sum_{k=1}^{N^{obs}} -\log Z[\Theta] - \sum_{i=1}^{N(V^{Or})(pg_k)} < \lambda_i^{(\alpha)}, H_i^{(\alpha)}(pg_k) > - \quad (4.4)$$

$$\sum_{j=1}^{N(R)(pg_k)} < \lambda_j^{(\beta)}, H_j^{(\beta)}(pg_k) >$$

$$(\Theta, R)^* = \operatorname{argmax}_{(\Theta, R)} \mathcal{L}(\Theta) . \quad (4.5)$$

Learning our parameters can then be broken down into two distinct stages:

1. Given a set of relationships R in the model, estimate $(\lambda^{(\alpha)}, \lambda^{(\beta)})$ for each Or node and each relationship.
2. Pursue a set of relationships R one-by-one to constrain the model.

Learning in this order may seem backward, but it is easier to understand the process if we first show the process for parameter estimation given a set of relationships R followed by the process for pursuing R .

4.2 Learning $(\lambda^{(\alpha)}, \lambda^{(\beta)})$

We solve for $\Theta = (\lambda^{(\alpha)}, \lambda^{(\beta)})$ using straightforward maximum likelihood estimation (MLE). Setting $\frac{\partial \mathcal{L}(\Theta)}{\partial \Theta} = 0$, we can solve for both sets of λ 's:

1. $\lambda^{(\alpha)}$: We have decided to treat each $\lambda_i^{(\alpha)}$ as into the summation to get $\mathcal{L}(\Theta) = \sum_{i=1}^{N(V^{Or})(pg_k)} - < \lambda_i^{(\alpha)}, \log H_i^{(\alpha)}(pg_k) >$. The derivative $\frac{\partial \mathcal{L}(\Theta)}{\partial \lambda^{(\alpha)}} = 0$ resolves to $\lambda_i^{(\alpha)} = -\log H_i^{(\alpha)}(PG^{obs})$. While we independent of other $\lambda_i^{(\alpha)}$'s. As such, we can move the $\log()$ in our likelihood function have discussed that we are modeling $H_i^{(\alpha)}$ as the

frequencies of occurrence of each Or node decomposition, we prove it more formally here.

The histogram $H_i^{(\alpha)}(PG^{obs})$ is the empirical distribution of all the possible Or node decomposition for Or node $v_i \in V^{Or}$. Thus,

$$H_i^{(\alpha)}(PG^{obs}) \approx p(\omega(v_i)) , \quad (4.6)$$

where $p(\omega(v_i))$ is the probability distribution over the switch variable $\omega(v_i)$ at node $v_i \in V^{Or}$. We can model our switch variables as simple multinomials. Letting θ_{ij} be the probability that $\omega(v_i)$ takes value j , and n_{ij} the number of times we observe this production, we can rewrite $p(\omega(v_i))$ as

$$p(\omega(v_i)) = \prod_{l=1}^{N(\omega(v_i))} \theta_{ij}^{n_{ij}} , \quad (4.7)$$

where $N(\omega(v_i))$ is the number of values $\omega(v_i)$ can take. We can plug this back into our prior and take the logarithm and derivative as before, this time with respect to our multinomial parameter θ_{ij} . Introducing Lagrange multipliers γ gives

$$\frac{\partial \mathcal{L}(\Theta)}{\partial \theta} = -N \frac{\partial \log Z[\Theta]}{\partial \theta} - \sum_{k=1}^{N^{obs}} \sum_{i=1}^{N(V^{Or})(pg_k)} \sum_{j=1}^{N(\omega(v_i))} \frac{n_{ij}^k}{\theta_{ij}} - \gamma = 0 \quad (4.8)$$

$$\gamma \left(\sum_{i=1}^{N(V^{Or})(pg_k)} \sum_{j=1}^{N(\omega(v_i))} \theta_{ij} - 1 \right) = 0 \quad (4.9)$$

Solving for the Lagrange multiplier and substituting into Equation 4.8 gives

$$\hat{\theta}_{ij} = \frac{-\sum_{k=1}^{N^{obs}} n_{ij}^k}{N_{\omega(v_i)} - N^{obs} \frac{\partial \log Z[\Theta]}{\partial \theta} + N^{obs} \frac{\partial \log Z[\Theta]}{\partial \theta}} = \frac{N_{ij}}{N_{\omega(v_i)}} \quad (4.10)$$

where $N_{\omega(v_i)}$ is the sum over all values of node v_i in all graphs in PG^{obs} . One can see that the estimator $\hat{\theta}$ is merely the sample frequency at each Or node. We can estimate the Or node probabilities by counting the number of times each Or node decomposes into each value divided by the number of times we see it decompose into anything.

Thus, each $H_i^{(\alpha)}$ is estimated simply by the counts of all possible $\omega(v_i)$ decompositions and, because $\lambda^{(\alpha)}$'s are independent of each other and of the $\lambda^{(\beta)}$'s, we can estimate them simply by

$$\lambda_i^{(\alpha)} = -\log H_i^{(\alpha)}(PG^{obs}) . \quad (4.11)$$

This is the MLE estimate for a multinomial and can be used to estimate production rule probabilities in grammars given that they are independent of any cross-link relations, i.e. context-free (Chi and Geman, 1998).

2. $\lambda^{(\beta)}$: Setting $\frac{\partial \mathcal{L}(\Theta)}{\partial \lambda^{(\beta)}} = 0$ yields

$$\frac{\partial \mathcal{L}(\Theta)}{\partial \lambda_j^{(\beta)}} = -\frac{1}{Z[\Theta]} \frac{\partial Z[\Theta]}{\partial \lambda_j^{(\beta)}} - E_f[H_j^{(\beta)}(PG)] \quad (4.12)$$

$$= E_p[H_j^{(\beta)}(PG)] - E_f[H_j^{(\beta)}(PG)] \quad (4.13)$$

which can be approximated as

$$\frac{\partial \mathcal{L}(\Theta)}{\partial \lambda_j^{(\beta)}} \approx H_j^{(\beta)}(PG^{syn}) - H_j^{(\beta)}(PG^{obs}) . \quad (4.14)$$

$H_j^{(\beta)}(PG^{syn})$ is the histogram formed from a set of parse graphs $PG^{syn} = \{pg_i^{syn} : i = 1, 2, \dots, N^{syn}\}$ that are synthesized from our current model $p(pg)$ for relationship r_j . The synthesized parse graphs are drawn by first sampling the number of children each node decomposes into according to the learned $\lambda^{(\alpha)}$ parameters. The appearances of the objects in the resulting parse tree are then Gibbs sampled according to the current $\lambda^{(\beta)}$ weights and the constraints in the model. The resulting parse graphs will be images, so we can compute histograms for the same relationship functions $f()$ over these parse graphs as we did over PG^{obs} . Thus $H_j^{(\beta)}(PG^{syn})$ is a measure of the statistics recreated from our model.

Solving for the $\lambda^{(\beta)}$'s such that $H_j^{(\beta)}(PG^{syn}) = H_j^{(\beta)}(PG^{obs})$; $\forall j$ can then be done using gradient ascent. We initialize the $\lambda_j^{(\beta)}$ weights to 0

$$\lambda_j^{(\beta)} = 0, j = 1, 2, \dots, N(R). \quad (4.15)$$

In the first stage when $\lambda_j^{(\beta)} = 0$, $H_j^{(\beta)}(PG^{syn})$ will be close to uniform. Gradient ascent is then used to update the $\lambda^{(\beta)}$'s,

$$\lambda_j^{(\beta)(t+1)} = \lambda_j^{(\beta)(t)} - \eta |H_j^{(\beta)}(PG^{syn}) - H_j^{(\beta)}(PG^{obs})|, \quad (4.16)$$

where η is a step factor that can depend on the iteration t and the distance between histograms can be measured using L_1 , L_2 , or another distance metric. This update reweights the $\lambda^{(\beta)}$'s for each histogram based on how much $H_j^{(\beta)}(PG^{syn})$ differs from $H_j^{(\beta)}(PG^{obs})$ for each relationship r_j . It reduces the energy for choosing underrepresented bins and increases the energy for choosing overrepresented bins during the next iteration of Gibbs sampling. After a number of iterations the $\lambda^{(\beta)}$'s will be weighted such that the synthesized distributions match the observed distributions, and thus $p(pg)$ will match $f(pg)$ along these dimensions. We run gradient ascent until all the statistics of the PG^{syn} sampled from our model match the observed statistics to within some distance ε_2 ,

$$|H_j^{(\beta)}(PG^{syn}) - H_j^{(\beta)}(PG^{obs})| < \varepsilon_2, \quad j = 1, 2, \dots, N(R). \quad (4.17)$$

Figure 4.1 shows a toy example of the $\lambda^{(\beta)}$ learning process. We begin with one observed histogram $H_j^{(\beta)}(PG^{obs})$ for relative car size and its corresponding $\lambda_j^{(\beta)}$ weight vector, which begins as a vector of all 0's. Because this weight is uniform, the images we sample in Step (2) look fairly random. In Step (3) we compute $H_j^{(\beta)}(PG^{obs})$, the distribution of relative car sizes across these sampled images. This, unsurprisingly, is fairly uniform as well because $\lambda_j^{(\beta)}$ was uniform. The figure shows these $H_j^{(\beta)}(PG^{syn})$ and $H_j^{(\beta)}(PG^{obs})$ superimposed below to emphasize the difference in their bin counts. In Step (4) we update the $\lambda_j^{(\beta)}$ weights according to how much each bin differs. Step (5) shows the sampled images resulting from the updated $\lambda_j^{(\beta)}$ weighting, which have much more appropriate relative car sizes. This process is carried out simultaneously for each $\lambda_j^{(\beta)}$ such that the statistics for every relationship in the model matches the observed statistics.

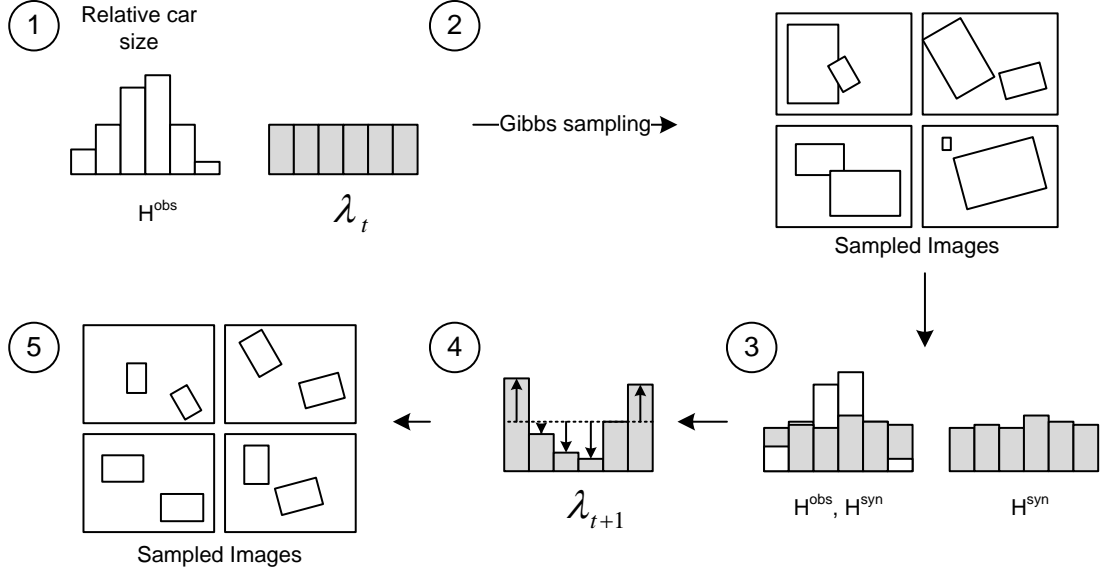


Figure 4.1: Examples of learning the relationship parameters, $\lambda_j^{(\beta)}$. (1) We begin with an observed histogram $H_j^{(\beta)}(PG^{obs})$, in this case the relative size between cars. $\lambda_j^{(\beta)}$ begins uniform. (2) Sampled images PG^{syn} are drawn from the model. (3) H^{syn} is computed for relative car size over the sampled images. (4) $\lambda_j^{(\beta)}$ is reweighted according to the difference between $H_j^{(\beta)}(PG^{obs})$ and $H_j^{(\beta)}(PG^{syn})$. (5) Newly sampled images appear scaled correctly.

4.3 Relationship Pursuit

The $\lambda^{(\beta)}$'s above were learned given that we already knew which relationships R existed in the model. We now show how to select the relationship constraints for the model. Because our dictionary of potential relationships Δ_R could be combinatorially huge, we will iteratively add relationships according to their importance instead of fitting the full model. Fitting the full model would require later attempts to sample from the model or perform inference with the model to check redundant relationships, making the model overly complex and slower to compute with.

We pursue the relationship set R by beginning with just an empty hierarchy,

$$p_0(pg; \Theta_0, R_0); R_0 = \{\emptyset\}. \quad (4.18)$$

This is the model with no parameters learned at all. We then learn the $\lambda^{(\alpha)}$'s, or tree parameters, using Equation 4.11, allowing us to sample images with the correct distributions of objects but without spatial or appearance constraints. Sampling the model at this stage would produce parse graphs with the correct number of objects, but without horizontal constraints, causing the resulting image to look more like an “alphabet soup” of objects that are big, small, overlapping, etc. We then iteratively add a new relationship r_+ from a dictionary of potential relationships Δ_R at each iteration to get a new distribution $p_+(pg; \Theta_+, R_+)$, $R_+ = R \cup \{r_+\}$. We choose r_+ such that we minimize $KL(f(pg)||p_+(pg; \Theta_+, R_+))$ at each step:

$$p_0(pg; \Theta_0, R_0) \rightarrow p_1(pg; \Theta_1, R_1) \rightarrow \dots \rightarrow p_k(pg; \Theta_k, R_k). \quad (4.19)$$

At each iteration we want to select the relation r_+ that brings our new model p_+ closest to f . We can measure how far our current model is from the target distribution f using the Kullback-Liebler (KL) divergence (Kullback and Leibler, 1951). We want to find the new p_+ that brings our model closest to the target distribution. Finding the p_+ that is closest to f is equivalent to finding the p_+ that brings our model the furthest away from the current model p . The relation r_+ that is maximally far away from p must also be maximally close to f and thus represents the largest decrease in KL divergence:

$$\begin{aligned} r_+ &= \operatorname{argmax}_r KL(f(pg)||p(pg; \Theta, R)) - KL(f(pg)||p_+(pg; \Theta_+, R_+)) \\ &= \operatorname{argmax}_r KL(p_+(pg; \Theta_+, R_+)||p(pg; \Theta, R)). \end{aligned} \quad (4.20)$$

See Appendix A for the derivation of Equation 4.20. One can see that minimizing the KL divergence between f and p_+ is equivalent to minimizing the entropy of p_+ , and must thus monotonically decrease at every iteration by adding new features.

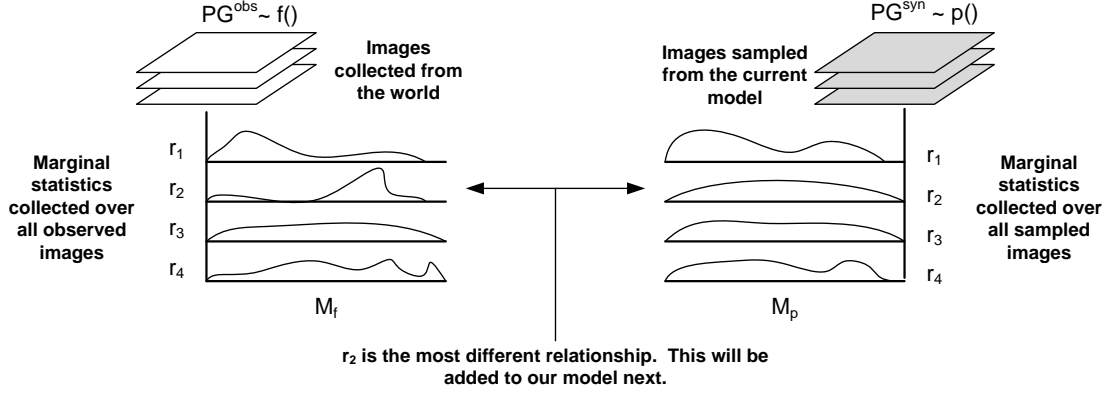


Figure 4.2: A visualization of the relationship pursuit procedure. The histograms over PG^{obs} are computed and compared against the histograms over PG^{syn} , which are sampled from the current model $p()$. The pair of histograms with largest Mahalanobis distance correspond to the relationship with most information gain r_+ that we should next add to the model.

The KL divergence is non-trivial to compute given that our model $p(pg; \Theta, R)$ has an intractable normalization constant in front of it and a different set of parameters from $p_+(pg; \Theta_+, R_+)$. Using a property of the relationship pursuit, however, we can approximate the decrease in KL divergence, otherwise known as the information gain, $\delta(r_+)$, using the Mahalanobis distance between the synthesized and observed histograms for the new potential relation r_+

$$\begin{aligned} \delta(r_+) &= KL(p_+(pg; \Theta_+, R_+) || p(pg; \Theta, R)) \\ &\approx d_{mahn}(H_+^{(\beta)}(PG^{obs}), H_+^{(\beta)}(PG^{syn})) . \end{aligned} \quad (4.21)$$

This holds due to a Taylor expansion around the relationship we're interested in adding, as shown in Appendix B.

In order to find the best new relationship to add to our model at a given iteration, we measure $H_j^{(\beta)}(PG^{obs})$ and $H_j^{(\beta)}(PG^{syn})$ for all relations $r_j \in \Delta_R$ and compare their Mahalanobis distances. The r_+ with the largest Mahalanobis between the syn-

thesized and observed histograms above some threshold is added to the model in the next iteration and its $\lambda_+^{(\beta)}$ parameters are learned as in the previous section. The pursuit ends when $\delta(r_+) < \varepsilon_1$. Figure 4.2 gives a high level visualization of the relationship pursuit process.

4.4 Summary of Parameter Learning and Relationship Pursuit Algorithms

The algorithm for learning the parameters of the model proceeds in two steps. We first learn the $\lambda^{(\alpha)}$'s by MLE, which are just the sample frequencies of the decompositions of each node. We then iteratively add spatial and appearance relations one-by-one until no relation remaining in Δ_R has Mahalanobis distance greater than ε_1 . After each relation is added, we iteratively update the $\lambda^{(\beta)}$'s for the current relation set to match $H_j^{(\beta)}(PG^{syn})$ to $H_j^{(\beta)}(PG^{obs})$, $\forall r_j \in R$. The algorithms are outlined below:

Algorithm 1. Relationship pursuit.

1. Begin with an empty model p_0 and observed parse graphs $PG^{obs} = \{pg_i^{obs} : i = 1, 2, \dots, N^{obs}\}$.
 2. Compute observed histograms $H_j^{(\beta)}(PG^{obs})$ and $H_j^{(\alpha)}(PG^{obs})$ for all relationships $r_j \in \Delta_R$ and all node frequencies for $v_i \in V^{Or}$, respectively.
 3. Approximate the $\lambda^{(\alpha)}$'s for the tree component using the sample frequencies from MLE.
 4. Repeat
 - (a) Sample N^{syn} parse graphs from the current model, $PG^{syn} = \{pg_i^{syn} : i = 1, 2, \dots, N^{syn}\}$.
 - (b) Calculate $H_j^{(\beta)}(PG^{syn})$, $r_j \in \Delta_R$.
 - (c) Select the relationship r_j for which $d_{manh}(H_j^{(\beta)}(PG^{syn}), H_j^{(\beta)}(PG^{obs}))$ is maximal as r_+ . Add r_+ to R .
 - (d) Relearn the $\lambda^{(\beta)}$'s for the new set $R_+ = R \cup \{r_+\}$ using Algorithm 2.
- until $d_{manh}(H_j^{(\beta)}(PG^{syn}), H_j^{(\beta)}(PG^{obs})) < \varepsilon_1, r_j \in \Delta_R$.

Algorithm 2. Parameter estimation algorithm.

1. Given a set of relations R and current model $p(pg; \Theta, R)$,
 2. Repeat
 - (a) Sample N^{syn} parse trees from the model, $PG^{syn} = \{pg_i^{syn} : i = 1, 2, \dots, N^{syn}\}$.
 - (b) Calculate $H_j^{(\beta)}(PG^{syn})$, $j = 1, 2, \dots, N(R)$.
 - (c) Update $\lambda_j^{(\beta)(t+1)} = \lambda_j^{(\beta)(t)} - \eta |H_j^{(\beta)}(PG^{syn}) - H_j^{(\beta)}(PG^{obs})|$ $j = 1, 2, \dots, N(R)$.
- until $|H_j^{(\beta)}(PG^{syn}) - H_j^{(\beta)}(PG^{obs})| < \varepsilon_2$, $j = 1, 2, \dots, N(R)$.

CHAPTER 5

Experiments on Learning and Sampling

To validate our learning algorithm for object modeling we present intermediate results of the learning process and show that samples drawn from our model are visually similar to the original training data, yet are completely novel unique objects that were never seen before by the model.

5.1 Experiments on Object Learning

In our first experiment we applied our algorithm to the task of learning the appearances of visual object categories.

5.1.1 Data Collection

To train our model we collected images for 24 different object categories. Each object category consisted of between 20 and 40 images taken under different poses and conditions. Each image consisted of an individual object. For each object category, we identified a set of archetypal parts (e.g. the teapot class consisted of (handle, base, lid, spout)) and labeled the corresponding object boundaries in each image.

5.1.2 Object Representation

Each object part is described by a boundary b that is defined as a graph,

$$b_i = \langle \nu_i, \zeta_i, l_i \rangle, \quad (5.1)$$

where ν is a set of boundary points and ζ is a set of edges, along with a label l_i indicating what type of part it is. The boundaries and labels are hand-labeled in every observed image I^{obs} . These objects form the bottom layer of the hierarchy. We create a node v_i in the contextual hierarchy for each boundary b_i so that every boundary is represented by a bottom-level node in the hierarchy and every bottom-level node in the hierarchy has a corresponding boundary representation.

Boundary Attributes

From the boundaries we can derive the appearance attributes $\phi(v_i) = \phi(b_i) = \{X_i, \theta_i, \sigma_i\}$ of each boundary b_i for each bottom-level node v_i in the hierarchy. The smallest enclosing bounding box, box_i , was computed for each b_i . The position of each part is its center of mass, its orientation is the major axis of box_i , and its scale is the length and width of box_i , treating the major axis as our measure of length and the minor axis our measure of width. If boundary b_i for node v_i consists of n vertices $\nu = \{\nu_1, \nu_2, \dots, \nu_n\}$ and $(left_i, right_i, top_i, bottom_i)$ describe the center points of the edges of b_i 's bounding box box_i , we can compute each part's appearance attributes as,

$$X_i = \left(\frac{\sum_{j=1}^n x(\nu_j)}{n}, \frac{\sum_{j=1}^n y(\nu_j)}{n} \right) \quad (5.2)$$

$$M_{axis} = \max((right_i - left_i), (top_i - bottom_i)) \quad (5.3)$$

$$m_{axis} = \min((right_i - left_i), (top_i - bottom_i)) \quad (5.4)$$

$$\theta = \cos^{-1}(M_{axis}, (1, 0)) \quad (5.5)$$

$$\sigma = (|M_{axis}|, |m_{axis}|) . \quad (5.6)$$

Figure 5.1 shows an example of the representation of an abstract part based on these

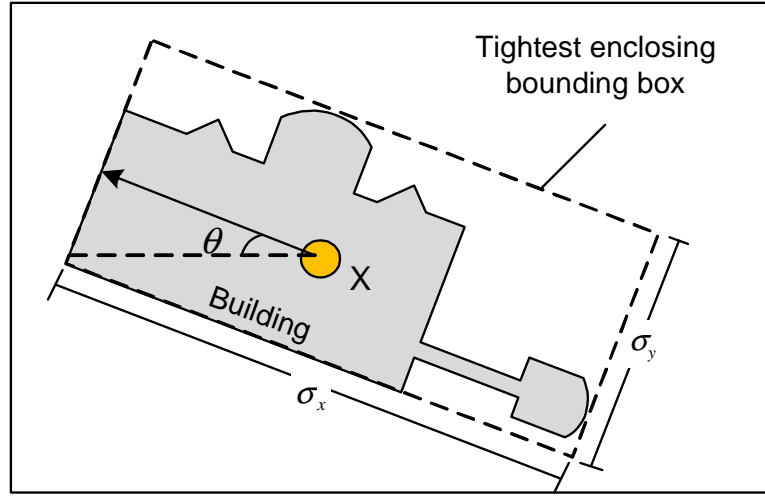


Figure 5.1: An example of the features computed for a single object. The boundary graph and smallest enclosing bounding box are used to compute position, scale, and orientation.

features.

Bonding Points

In order to make sure that certain object parts link together smoothly, we select a subset of the boundary points $z_i \subseteq \nu_i$ that are common across object boundaries of the same type to act as bonding points. These bonding points represent the template points of each part, and attempt to capture the salient defining points for each type of boundary. Figure 5.2 shows examples of common bonding points for three types of parts. The bottom of each teapot spout, regardless of the rest of its shape, is indexed by the same type of bonding point. These templates of bonding points allow us to measure tighter relationships between object parts, as we can focus on the relationships between specific shape points, instead of amorphous common points, like centroids, which may vary drastically depending on the shape of each part.

All boundaries of the same part type p_j (e.g. all clock hands) can be grouped into

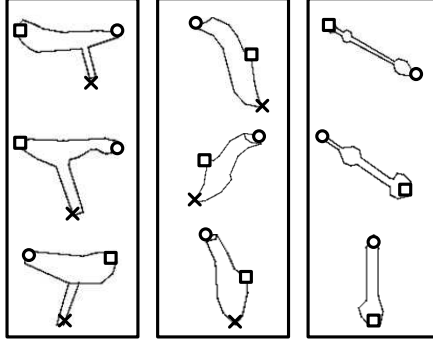


Figure 5.2: Examples of corresponding bonding points for each terminal type.

the set B^j ,

$$B^j = \cup b_i; \forall l_i = j \quad (5.7)$$

For each part category B^j we created a set of bonding points BP_j that define key shape features, such as high curvature.

$$BP_j = \{bp_{jk}; k = 1, 2, \dots, n\} . \quad (5.8)$$

These bonding points define a template of keypoints for a specific part type j . Each boundary of the same type $b_i \in B^j$ may have a varying number of points on its boundary, so we must define a mapping M from BP_j to each boundary's specific points:

$$z_i = M(j, k, b_i) = \nu_i(bp_{jk}) . \quad (5.9)$$

In other words, for each bonding point in the template set BP_j , we have an index to a point in each template b_i with label $l_i = j$ that corresponds to that point.

The bonding points are determined automatically by matching every shape $b_i \in B^j$ to the first boundary in that set $b_0 \in B^j$ using shape context (Belongie et al., 2002). Any points that are matched across all instances are retained as bonding points and their indices recorded in the mapping function $M()$.




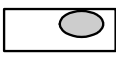

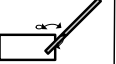


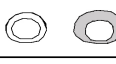

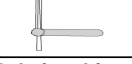
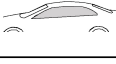



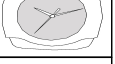
Soft Relationships			Hard Relationships				
Position	Scale	Orientation	Contained	Hinged	Attached	Butting	Concentric
							
							
Low Level Relationships			High Level Relationships				

Figure 5.3: Examples of relationships between object parts that could exist in our model.

5.1.3 Relationship Functions

Figure 5.3 shows the relationship constraints that we model between object parts. The relative position, orientation, and scale of object parts is critical for learning the appearances of different objects. We also include higher level relationships such as containment, concentricity, and whether two parts abut. With these relationships, we can capture statistical constraints between the object parts so that we can sample new instances of that object class and recognize instances in new images. In this work we will focus mostly on low level relationships. All of our images used for training taken under similar positional conditions, i.e. each photograph is taken from ground-level, not overhead, so there are strong correlations between relative part positions in the X and Y directions individually. We therefore model scale and position relationships in X and Y independently. Table 5.1 shows the equations used to compute the relationship values between pairs of parts.

5.1.4 Parse Graph Construction

We define our grammar such that there are no ambiguous parse graphs. This is simple to achieve simply by ensuring that no node has more than one parent. We can then generate a parse graph for an image just from the labeled parts of the object, as in

Table 5.1: Object relationship function definitions.

Relationship	n Nodes	Function $f_i()$
Aspect ratio	1	σ_y/σ_x
Position X	2	$(X_{x2} - X_{x1})/\sigma_{x1}$
Position Y	2	$(X_{y2} - X_{y1})/\sigma_{y1}$
Scale X	2	σ_{x1}/σ_{x2}
Scale Y	2	σ_{y1}/σ_{y2}
Relative orientation	2	$\theta_2 - \theta_1$
Contained	2	$Area_{overlap}/Area_1 = 1$
Concentricity	2	$\sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$
Butting	2	$ X_2 - X_1 < \epsilon_1 \& \theta_2 - \theta_1 - 90 < \epsilon_2$

Figure 3.5(b). Each labeled part in an image maps to a node in the contextual hierarchy and we derive the parse tree as the tree formed by the nodes of the present. Once the tree is determined any relationships can be measured between pairs or sets of object parts in the tree to form the horizontal edges.

5.1.5 Learning the $\lambda()$ Parameters

Histogram Representation For Relationships

We have chosen to represent our relationship statistic distributions as histograms. This is intended to save us the trouble of fitting specific distributions to each new relationship. In order to use histograms, we made the following design decisions:

1. The range of the histograms are determined by the maximum and minimum values observed in the training data. We model values outside of this range by a decreasing gradient function. Define the maximum probability value allowed

for this gradient function as p and the width of the histogram bin as w . Then a point that is distance k bin widths beyond the edge of the histogram (i.e. $k * w$ distance from the edge) can be assigned probability $p - (k/n) * p$, where n is as many extra bins as we'd like to add to either side of our histogram. Thus, values that are beyond the histogram edge are assigned a probability that is a fraction of the maximum probability p , depending on how far away they are. n is usually set to be something large, e.g. 10000, and p is usually set to be the edge bin probability, which prevents the tail from being greater probability than the probability in the edgemost bins. Values greater than n are assigned probability 0 (or some minimum probability). These tails allow us to model values outside of the histogram range while guaranteeing that their probability is never greater than the probability for the edge closest to that side.

2. We divide our histograms into 10 bins each. We were surprised to find that, empirically, any number of bins above 6-7 were sufficient to produce samples from the model that are perceptually similar to real images. Obviously we will never perfectly match the distributions with this discretization until we approach the limit, but we find suitable results with even as few as 10 bins.

Results of Histogram Learning

Figure 5.4 shows a sample of five relationship histograms learned by our model for the “teapot” object class, Scale X, Scale Y, Position X, Position Y, and butting. The dark black line shows the true observed distribution for each relationship, which is ultimately the statistic we would like to recreate. The dotted blue line shows the recreated distributions from our model after the first iteration. We can see that our recreated distributions are not at all close to the true statistics from the real world before the $\lambda()$ parameters are learned. By the final iteration, however, we have matched the true distributions almost exactly, as shown by the red dotted lines. These plots show that,

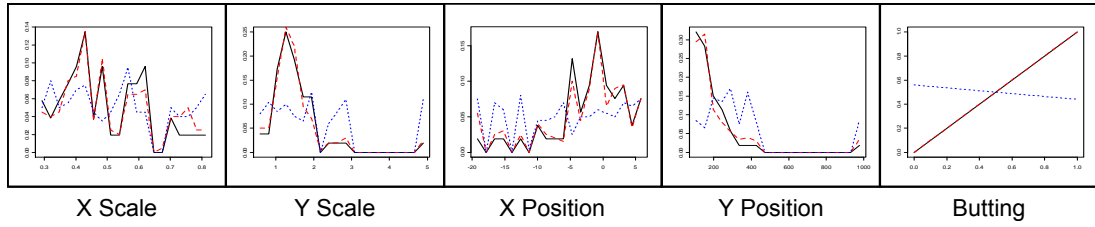


Figure 5.4: Histograms for five relationships for the teapot class during the learning process.

given a set of relationships R , our learning algorithm can learn weight vectors that accurately reweight the distributions until our model recreates the real world statistics for the object class. Because our generated relationship distributions match the true relationship distributions, we know that our model of objects must look similar to true objects, at least along those dimensions. Thus, our model has captured what is most salient about object categories and recreated it.

5.1.6 Relationship Pursuit

One way to test the validity of our model is to draw samples from it as in Section 4.2 and see how visually similar samples from our model are to real world objects. Figure 5.5 shows samples drawn from the bike and clock categories at various iterations of the relationship pursuit. We can see that, during the early iterations, when we’ve only added a few relationships, the scale, positions, and orientations of the part are seemingly random. As we add more relationships, however, the objects become increasingly coherent until, by the end of the relationship pursuit, the objects that we observe are similar to real world objects. To the right of each row is a plot of the Kullback-Liebler (KL) divergence between the model at each iteration and the true distribution. We can see that the first few relationships reduce the KL divergence most significantly. Toward the end of the relationship pursuit the algorithm adds relation-

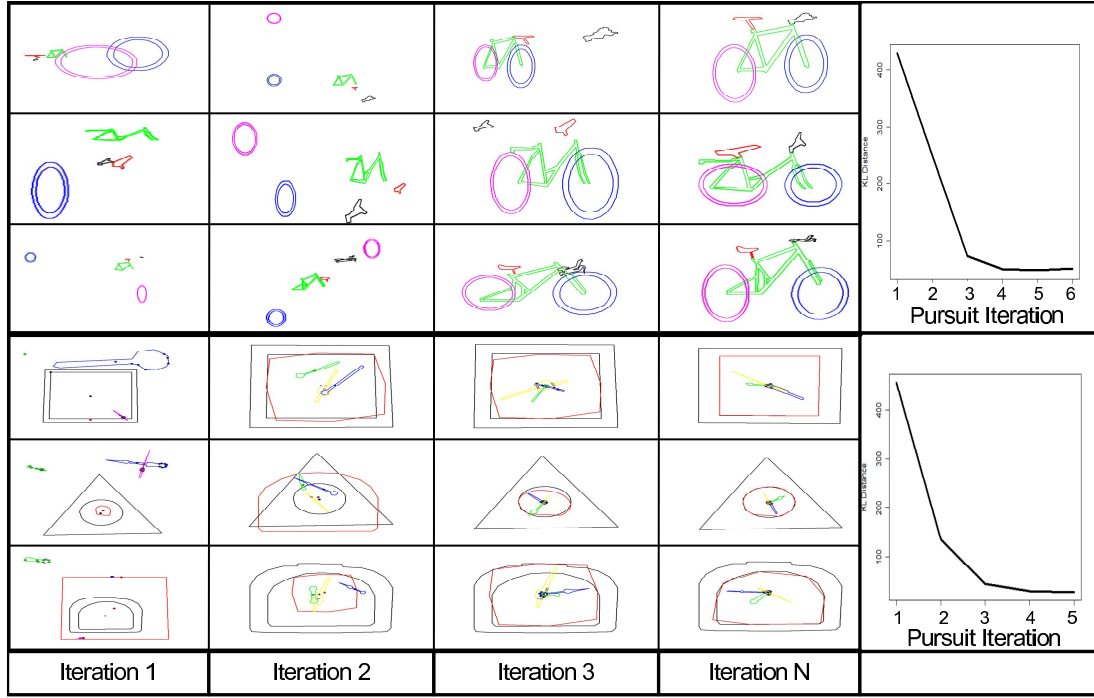


Figure 5.5: Examples of clocks and bicycles drawn from our model during different stages of the relationship pursuit. Estimated Kullback-Liebler divergence between the target distribution and the current model is shown on the right as measured by the information gain at each iteration.

ships with lower information gain. We can see that the KL divergence is monotonically decreasing over time.

5.1.7 Analysis by Synthesis

Figure 5.6 shows sampled objects from 24 categories of objects alongside original images from that class. The outlined objects are generated by sampling object parts according to their learned relationships. We can see that the resulting objects are very similar to the object class from which they originate even though they are completely novel instances of each object. The realism of each object category verifies that the

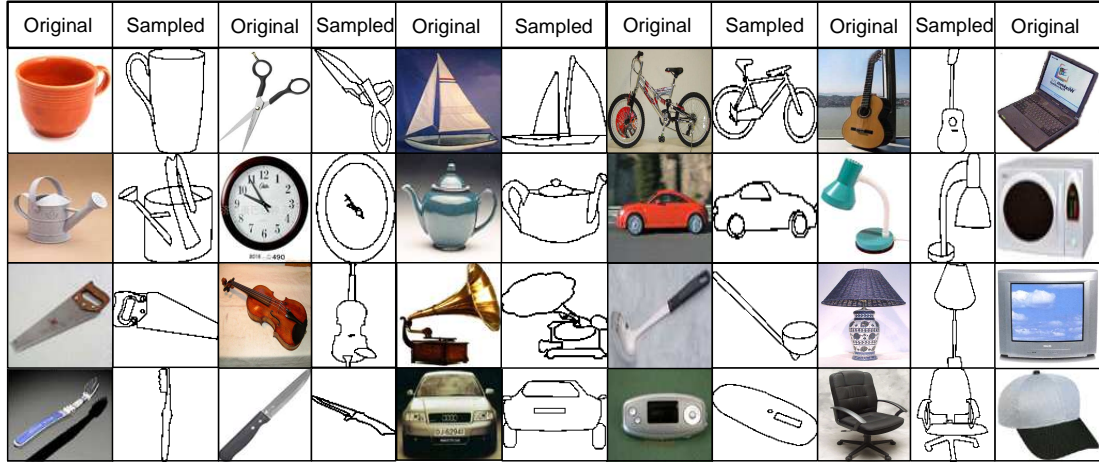


Figure 5.6: Examples of objects sampled from our model for 24 different object categories. The samples selected look very similar to the original images, though they are novel instances.

model is capturing the relevant statistics for each object category.

5.1.8 Small Sample Set Generalization

One of the main features of our model is its ability to learn representations from a small sample set as well as its ability to generalize to a combinatorial number of novel instances. Thus, we can learn the model from a small training set, yet still recognize objects in a testing set that were never seen before. To demonstrate this ability, we first show the minimum training size needed to learn a category. In this experiment, we divide each category’s data into a training and testing set, Ω^{train} , Ω^{test} . For a given k , $0 < k \leq N$, we determine the subset of size $k \in \Omega^{train}$ that produces the most coverage of the testing set. Coverage is measured by drawing N samples $\Omega^{sample} = \{s_1, s_2, \dots, s_N\}$ from the model learned from the subset k and measuring

$$c = \frac{\sum_{i=1}^N 1(s_i \in \Omega^{test})}{|\Omega^{test}|} \quad (5.10)$$

i.e. what percentage of the test set appeared in our sample set. For an instance in the sample set and the test set to be considered equal, they must be composed of the same types of leaf nodes and satisfy the relational constraints of the model. In effect we are measuring how many instances we need to learn in order to be able to recreate all the instances in both the training and testing set. The optimal subset is the smallest subset of training examples from which we can learn a model that recreates all the instances in the class.

To determine the optimal subset of size k , we begin with an arbitrary k samples and compute c . We then randomly swap an instance from the sample set with another instance from the training set, relearn the model, then recompute c' . If $c' > c$, we accept the change, otherwise we revert to our old sample set. This continues for $n = 1000$ iterations. In effect we are probabilistically searching for the subset of size k that creates the best coverage for each value of k .

Figure 5.7 shows the coverage results for a sample of categories as k increases. We can see that very few training instances are required to learn a model with all the expressability of one learned with many. This implies we may need very few instances in an unsupervised framework to learn the maximally powerful model. Obviously there are some instances with parts that we've never seen in the testing set, so it may not be possible for each category to reach 100% coverage. Were we to unite the training and testing set, we could cover these outliers as well.

To demonstrate the generalizability of the model, Figure 5.8 shows the smallest subset for the lamp category that achieved maximal coverage, along with instances drawn from the learned model. Only 6 training samples are needed, yet their parts can be reconfigured and adjusted according to the model to produce radically different instances from the training set. Note that the part configurations and appearances in the samples differ greatly from those in the training set, yet the objects are still coherent.

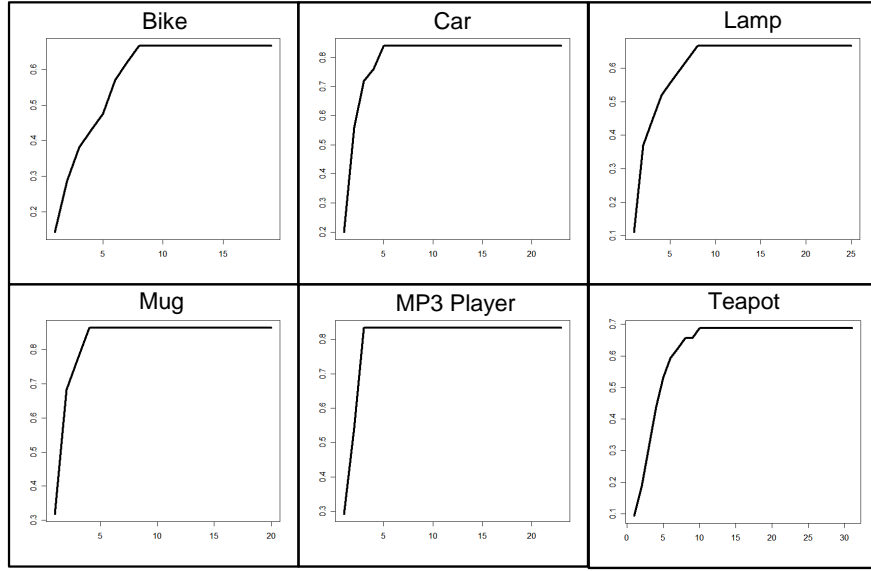


Figure 5.7: Coverage results for 6 categories showing number of reproducible instances against training size. Only a small set of data is needed to learn the most expressive model.

This is useful for recognition tasks, where new instances can be recognized despite not appearing in the training data. One can also generate large amounts of training data for discriminative tasks using this model, learned from a small, easily obtained set of images.

5.2 Learning Results for Aerial Images

We also tested our learning algorithm for aerial image modeling. We deterministically constructed parse graphs from labeled images and then were able to show convergence of the relationship histograms and realistic sampling results.

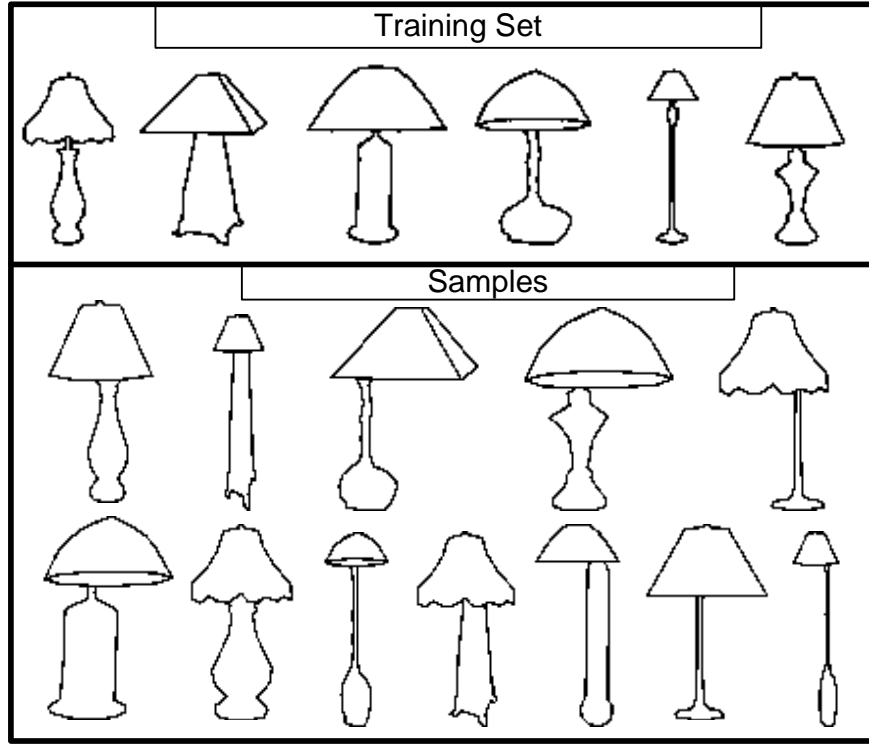


Figure 5.8: Demonstration of the model’s generalizability. The model learned from only 6 training instances can produce the varied samples below.

5.2.1 Data Collection

We selected 120 aerial images from the Lotus Hill Database (Yao et al., 2007), which included labeled boundaries of roofs, roads, parking lots, tree regions, and cars to use as our training data ¹ These boundaries are hand-labeled. The images ranged in size from 640x480 pixels to 1000x1000 pixels.

5.2.2 Object Representation

We represent our five object categories, (roads, trees, roofs, cars, parking lots), by their enclosing boundaries, just as in Section 5.1.2. We do not need bonding points for aerial

¹Dataset available from <http://www.imageparsing.com>. More data will be released after the publication of this dissertation, but sample data is available free for downloading now.

Table 5.2: Relationship function definitions.

Relationship	n Nodes	Function $f_i()$
Aspect ratio	1	σ_y/σ_x
Relative position	2	$(X_2 - X_1)/s_1$
Radial position	2	$\{ X_2 - X_1 , \theta(X_2 - X_1) - \theta_1 \}$
Relative scale	2	s_1/s_2
Relative orientation	2	$\theta_2 - \theta_1$
Percentage overlap	2	$Area_{overlap}/Area_1$
Alignment	n	SSE of least squares fit

images, however, as the parts we are modeling are individual objects that do not need to be tightly bonded.

5.2.3 Relationship Functions

The functions f_i for each relationship are defined over the attributes of sets of nodes, $\phi(V_i)$. We implemented the relationship functions listed in Table 5.2. Relative position returns the vector between the centers of the two objects, which is relative to the coordinate frame of the image. This is not particularly useful as aerial images rarely have a well-defined “top” or “bottom”. Similarly we do not use Position X/Y or Scale X/Y relationships as we did for objects, as the X and Y coordinate frame is not fixed when we are modeling overhead images. Radial position attempts to deal with this by measuring relative distance in polar coordinates. Figure 5.9 shows a visualization of some of the relationships we measure between objects.

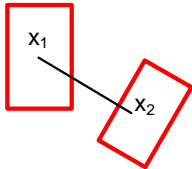
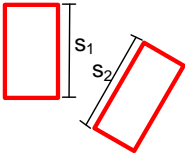
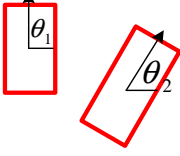
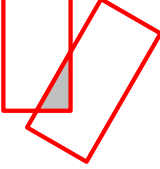
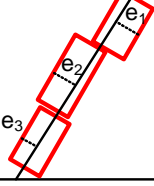
Relative Position	Relative Scale	Relative Orientation	Overlap	Alignment
				

Figure 5.9: Visualizations of inter-object relationships used during aerial image modeling.

5.2.4 Deterministically Forming Parse Graphs

Each I^{obs} has a corresponding parse graph pg^{obs} describing the hierarchical arrangement of its objects. Unlike object parse graphs, which tend to have a small number of specific leaf nodes, we need to determine higher-level group nodes for the aerial image case.

We are given the boundaries of every object as described in Section 5.2.2. To form parse graphs from a collection of labeled objects (the leaf nodes) we make the following stipulation:

Proposition 1 Boundaries within distance ϱ of each other that are of the same object label will be considered members of the same group.

In other words, objects are deterministically assigned to groups according to their distance between one another. This provides two benefits:

- 1) We can deterministically form a hierarchy from a flat set of objects.
- 2) We only measure relationship statistics within and between groups of objects, so limiting the distance at which two objects are related prevents us from calculating and learning statistics of objects that are very far away.

In our experiments we set ϱ to be label-dependent. If we let s be b_i 's aspect ratio, we can define a set of distance thresholds as in Table 5.3. For example, a tree would

Table 5.3: Category-dependent distance thresholds ϱ for deterministic grouping based on an object’s aspect ratio s .

Object Label	ϱ
Roof(s)	s
Car(s)	$0.5*s$
Tree(s)	$1.2*s$
Road(s)	$0.5*s$
Parking Lot(s)	s

need to be within 1.2 of its aspect ratio of another tree to be considered in the same group. Because some objects within the same label may vary significantly in size, one may also choose to consider two objects proximal only if they are within $c*\min(s_1, s_2)$ of each other, where (s_1, s_2) are the aspect ratios of the two objects in question. This is particularly useful when determining which groups of objects should be associated, as their sizes can vary significantly more than those of single objects.

Figure 5.10 shows an example of deterministically forming a parse graph from a set of labeled objects. In this example, cars that are nearby one another are grouped together, as shown in (a). The same goes for roofs labeled in (a). Figure 5.10(b) shows the resulting groups from this first step and their distance-based relationships as well.

An important point to note here is that, though we form parse graphs deterministically for our observed images, we do **not** form them deterministically when inferring the best explanation of a new image. In our training images I^{obs} we are making the assumption that proximal objects are grouped and that this grouping defines the number of objects that the group consists of (decomposes into). There may, however, be proximal groups in our testing images that have, for example, a different number of



(a) Network structure between single objects



(b) Network structure between grouped objects

Object/Group boundary
 Object/Group center
 - - - - - Neighborhood Edge

Figure 5.10: An example of deterministically forming the neighborhood structure for a parse graph from labeled objects. (a) Cars and roofs that are within a certain distance of each other are grouped together. (b) Groups that are within a certain distance of each other have group-level constraints applied.

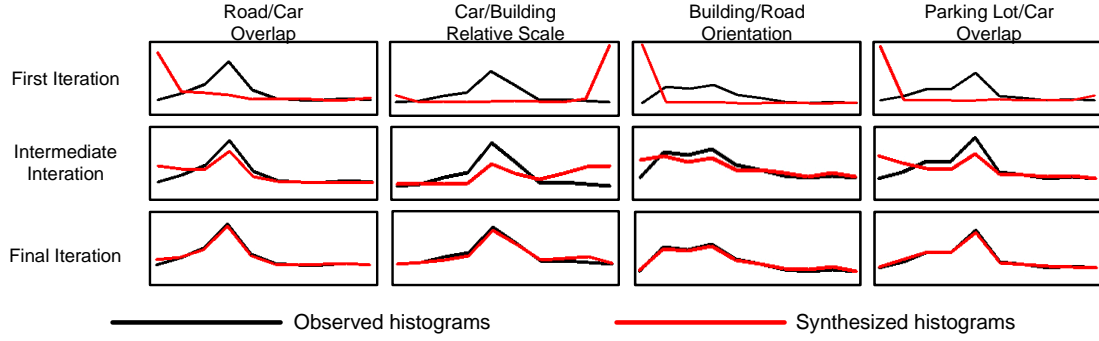


Figure 5.11: Histograms for four typical relations over the course of the learning algorithm. The black lines are the histograms of the observed data, $H^{(\beta)}(PG^{obs})$, and the red lines are the histograms of the synthesized data, $H^{(\beta)}(PG^{syn})$, at each iteration. At first the statistics of the synthesized data are so far off from the truth that most values are out of bounds. Halfway through the learning process the histograms look close to matching and by the final iteration the histograms match nearly perfectly.

objects than we expect to see based on our training data. In this case, it may make more sense to split the group into subgroups that match our learned decomposition frequencies than grouping them all under a hard-coded proximity condition. In a world where we’ve only seen sets of three cars, a row of six cars is more consistently explained as two sets of three by our model.

5.2.5 Learning the $\lambda()$ Parameters

We use a histogram representation as we did in Section 5.1.5. We use 10 bins again and a decreasing gradient function at the ends of the histogram to represent vanishing probabilities in the tails.

We set $\varepsilon_1 = 4$ and $\varepsilon_2 = 0.2$ for Algorithms 1 and 2 and then learn a hierarchical contextual model of objects in aerial images. Figure 5.11 shows $H^{(\beta)}(PG^{obs})$ and $H^{(\beta)}(PG^{syn})$ for four typical relations at three different iterations of the parameter

learning algorithm. In the first iteration, the histograms from our synthesized images are so far away from the true histograms that most of their data is out of bounds. Halfway through the learning, however, the histograms start to look coarsely similar. By the final iteration, the histograms have matched nearly perfectly. This assures us that the $\lambda^{(\beta)}$'s are reweighting the histogram bins correctly such that, over time, the images we synthesize using our model match the statistics of true aerial images.

We used 5 object categories in our model (car, roof, road, parking lot, tree) and their 5 corresponding group categories. We used 7 relationship functions in our model, resulting in a relationship dictionary Δ_R consisted of 360 possible relationships (10 aspect ratio relations + (5 objects)*(5 objects)*(7 relationships) + (5 groups)*(5 groups)*(7 relationships)). Of those 360 possible relationships our model selected 27, consisting mostly of overlap relations (car/car overlap, building/tree overlap, car/parking lot overlap), relative scale relations (car/car relative scale, roof/road relative scale), and alignment relations (car/car alignment). There were also a few orientation relations added, though they were only slightly better than noise (roof/road orientation) and could probably be weeded out by adjusting ϵ_1 .

5.2.6 Analysis by Synthesis

Figure 5.12 shows samples from our final model for aerial images $p(pg; \Theta, R)$. The resulting images appear similar to true aerial images, with objects obeying many of the same spatial and appearance constraints that we observe in the real data. We see cars appearing on roads, roofs arranged in blocks, and few or no spurious overlaps. Note that these samples are not representative of a specific aerial image from the training data or elsewhere. These are simply object boundaries that have been scaled, positioned, and oriented such that they minimize the energy in our prior. Nevertheless, we see that the relationship histograms match between the two models and the sam-

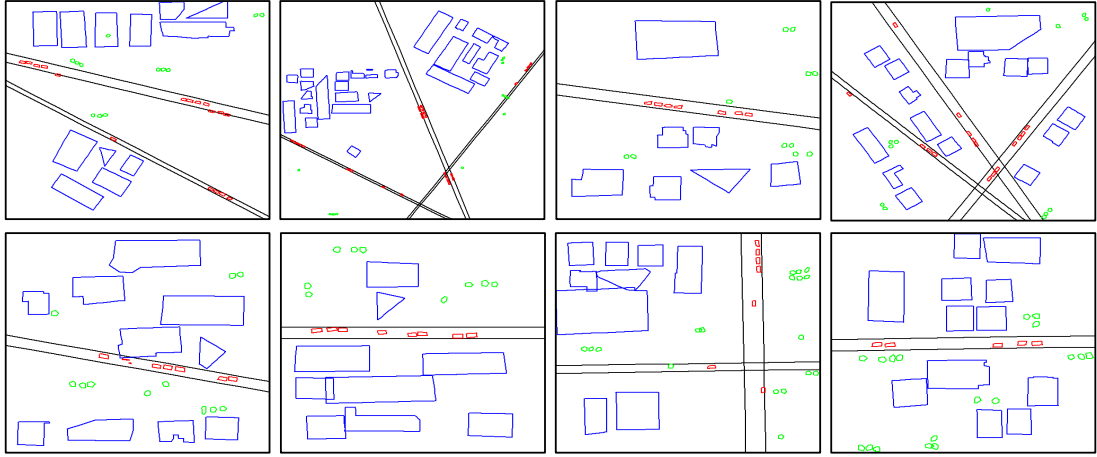


Figure 5.12: Samples from our learned aerial image model (blue = roofs, red = cars, black = roads, green = trees). These are not images directly sampled from the training data, but collections of objects obeying the statistics of our learned model. We can create a vast amount of unique object configurations even though we’ve never observed them directly.

pled images are perceptually similar to true aerial images. This shows that our learned model is in fact capturing the relationship statistics present in true aerial images and can thus recreate believable aerial image configurations.

5.3 Conclusions on Learning Experiments

We are very satisfied with the results of the learning algorithm for object modeling and aerial image modeling. The histograms drawn from the model during learning show that incremental updates of the $\lambda()$ parameters enable the model to recreate the expected statistics from the real world. In addition, the model is selecting an appropriate set of relationships and learning their parameters correctly, as evidenced by the samples drawn from the model. We can see in the generated images that the relevant patterns and appearances are being learned and that the model has captured the

essence of the patterns it has learned.

CHAPTER 6

Inference With The C^4 Algorithm

We must now tackle the difficult task of performing inference with the models learned in Chapter 4. Given a set of part detections, whether these are object parts or individual objects in an aerial image, our goal is to determine which subset of those parts, if any, correspond to true objects and object parts. We present a novel probabilistic cluster sampling algorithm, C^4 , to efficiently find the subsets of parts that most likely correspond to true detections. We will outline the motivation for this new inference algorithm and then give its formal definition. We will conclude the chapter with an analysis of C^4 's performance on graph labeling and subset selection problems and then show an extension to a hierarchical formulation for improved performance.

6.1 Introduction

We will begin with an introduction to the inference problem we are trying to solve and a look at why conventional inference methods may not be the optimal choice for our type of model.

6.1.1 Motivations and Objective

Many vision tasks, such as scene labeling (Kumar and Hebert, 2003; Porway et al., 2008; Rosenfeld et al., 1976), object detection/recognition (Felzenszwalb and Hutten-

locher, 2005; Torralba et al., 2004), segmentation (Cormen et al., 1988; Tu and Zhu, 2002), and graph matching (Chui and Rangarajan, 2003; Lin et al., 2009) are formulated as energy minimization (or maximum a posteriori probability) problems defined on graphical models – Markov random fields (Besag, 1986; Geman and Geman, 1984), conditional random fields (Kumar and Hebert, 2003; Lafferty et al., 2001), or hierarchical graphs (et al., 2004; Zhu and Mumford, 2006). These optimization problems become exceedingly difficult when there are multiple solutions, i.e. distinct modes, with high probabilities and in some cases equal probability.

Figure 6.1 shows examples of typical scenarios that have multiple, equally likely solutions in the absence of further context. The top row shows the well-known Necker Cube which has two valid 3D interpretations. The middle row is the Wittgenstein illusion, in which the drawing can appear to be either a duck or a rabbit. Without further context, we cannot determine the correct labeling. The bottom row shows an aerial image for scene labeling. It can be explained as either a roof with vents or a parking lot containing cars.

Computing multiple solutions is important for preserving the intrinsic ambiguities and avoiding early commitment to a single solution which, even if it's currently the globally optimal one, may turn out to be less favorable when later context arrives. However, it is a persistent challenge to enable algorithms to climb out of local optima and to jump between solutions far apart in the state space. Popular energy minimization algorithms, such as Iterative Conditional Modes (ICM) (Besag, 1986), Loopy Belief Propagation (LBP) (Kumar and Torr, 2006; Weiss, 2000), and graph cuts (Boykov et al., 2001; Kolmogorov and Rother, 2007) compute one solution and thus do not address this problem. Existing MCMC algorithms, such as various Gibbs samplers (Geman and Geman, 1984; Liu et al., 1995), DDMCMC (Tu and Zhu, 2002), and Swendsen-Wang cuts (Barbu and Zhu, 2005; Swendsen and Wang, 1987), promise

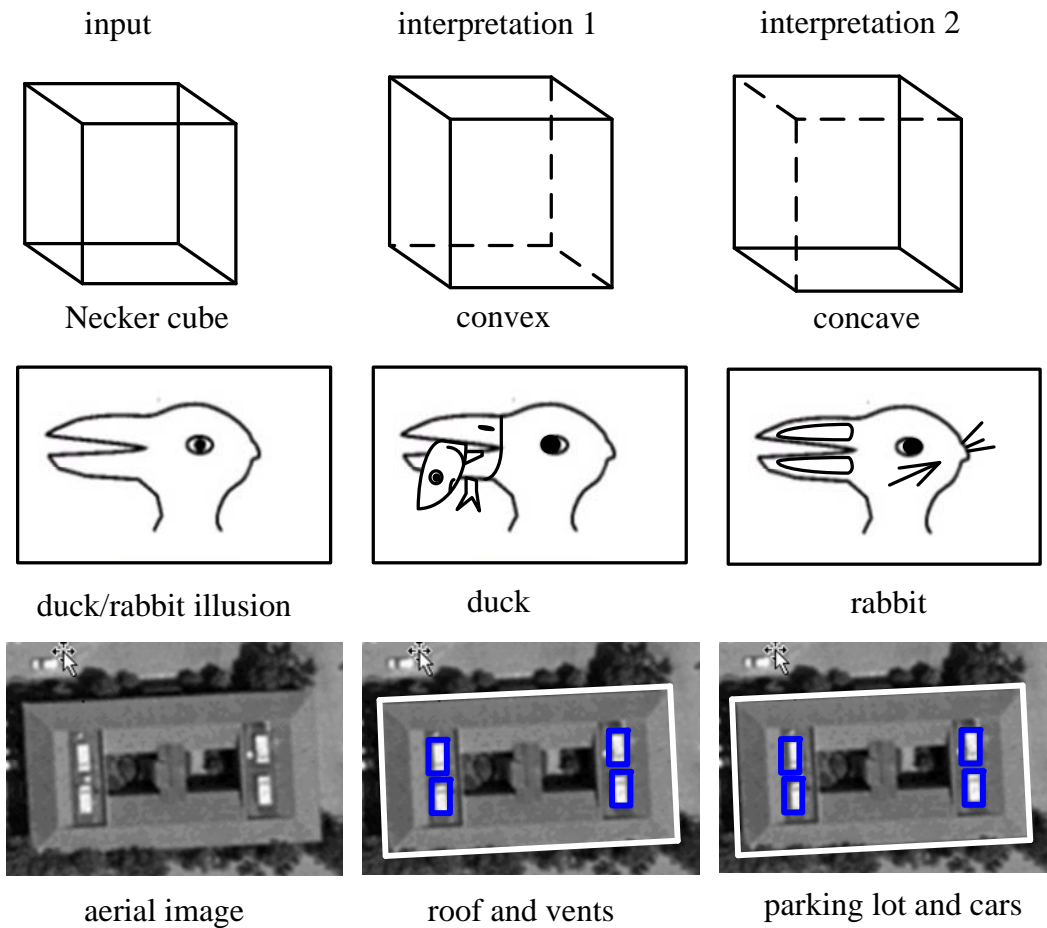


Figure 6.1: Examples of problems with multiple solutions: (top) the Necker Cube; (Middle) the Wittgenstein illusion; and (Bottom) An aerial image interpreted as either a roof with vents or a parking lot with cars. Ambiguities should be preserved until further context arrives.

global optimization and ergodicity in the state space, but often need long waiting time in moving between distinct modes, which needs a sequence of lucky moves up the energy landscape before it goes down.

Our objective is to develop an algorithm that can compute multiple solutions and thus preserve the ambiguities on rather general settings:

1. The graph can be flat, such as MRF or CRF, or hierarchical, such as a parse graph.
2. The graph may have positive (cooperative) and negative (competitive or conflicting) edges for both hard or soft constraints.
3. The probability (energy) defined on the graph is quite general, even with energy terms involving more than two nodes.

In vision, it is safe to assume that the graph is locally connected and we do not consider the worst case scenario where graphs are fully connected.

6.1.2 Related Work in the Literature

In the 1970s, many problems, including line drawing interpretation and scene labeling, were posed as constraint satisfaction problems (CSPs). The CSPs were either solved by heuristic search methods (Pearl, 1984) or constraint propagation methods (Apt, 1999; Mackworth, 1977). The former keeps a list of open nodes for plausible alternatives and can backtrack to explore multiple solutions. However, the open list can become too long to maintain when the graph is large. The latter iteratively updates the labels of nodes based on their neighbors. One well-known constraint propagation algorithm is the relaxation labeling method by Rosenfeld, Hummel, and Zucker in 1976 (Rosenfeld et al., 1976).

In the 1980s, the famous Gibbs sampler – a probabilistic version of relaxation labeling – was presented by Geman and Geman in 1984 (Geman and Geman, 1984). The update of labels is justified in a solid MCMC and MRF framework and thus is guaranteed to sample from the posterior probabilities. In special cases, the Gibbs sampler is equal to belief propagation (Pearl, 1984) for polytrees and to dynamic programming in chains. The Gibbs sampler is found to slow down critically when a number of nodes in the graph are strongly coupled. Figure 6.2 illustrates an example using the Necker Cube where the six internal lines are divided into two coupling groups: (1-2-3) and (4-5-6). Lines in each group must have the same label (concave or convex) to be valid as they share the two 'Y'-junctions. Thus, updating the label of a single line in a coupled group does not move at all, unless we update the label of the whole group together, i.e. all six labels in one step. The problem is that we don't know which nodes in the graph are coupled and to what extent they are coupled for general problems with large graphs. In 1987, a breakthrough came from two physicists, Swendsen and Wang (Swendsen and Wang, 1987), who proposed a cluster sampling technique. The Swendsen-Wang (SW) method finds coupled groups, called “clusters”, dynamically by turning the edges in the graph on/off according to the probabilities defined on these edges. The edge probability measures the coupling strengths. Unfortunately, their algorithm only works for the Ising and Potts models. We will discuss the SW method in later sections.

The 1990s were relatively quiet. There were numerous attempts made to improve MCMC methods (see Liu (Liu, 2001) for surveys), such as the block Gibbs sampler (Liu et al., 1995), which produced little improvement. Green formulated reversible jumps in 1995 (Green, 1995) following the jump-diffusion algorithm by Grenander and Miller in 1994 (Grenander and Miler, 1994). In 1999, Cooper and Frieze analyzed the convergence speed of SW using a path coupling technique and showed that the SW method has a polynomial mixing time when the nodes in the graph are connected to a

constant number of neighbors (Cooper and Frieze, 1999).

In the 2000s, a few non-MCMC methods generated remarkable impacts on the vision community. For example, the loopy belief propagation (LBP) algorithm by Weiss et. al. (Weiss, 2000) and the graph cut algorithms by Boykov, Kolmogorov, et. al. (Boykov et al., 2001; Kolmogorov and Rother, 2007). These algorithms are very fast and work well on special class of graph structures and energy functions, but they do not address the general problems, such as computing multiple solutions. On the MCMC side, Tu and Zhu developed the Data-Driven Markov Chain Monte Carlo (DDMCMC) algorithm for image segmentation in 2002 (Tu and Zhu, 2002), which uses bottom-up discriminative probabilities to drive the Markov chain moves. They also developed a “K-adventurer” procedure to keep multiple solutions. The DDMCMC method was also used by Dellaert (Oh et al., 2005) for tracking bee dances. Dellaert also used MCMC to explore correspondences for structure-from-motion problems, even incorporating a “jump parameter” to allow the algorithm to jump to new solutions (Dellaert et al., 2001). In 2005, Barbu and Zhu proposed the SW-cut algorithm (Barbu and Zhu, 2005) which, for the first time, generalized the SW method to arbitrary probabilities models. As we will discuss in later sections, the SW-cut did not consider negative edges, high order constraints, or hierarchical graphs and is less effective in swapping between competing solutions. The C^4 algorithm in this chapter is a direct generalization of the SW-cut algorithm (Barbu and Zhu, 2005).

6.1.3 Overview of the major concepts of C^4

In this chapter we present a probabilistic clustering algorithm called *Clustering Cooperative and Competitive Constraints* (C^4) for computing multiple solutions in graphical models. We consider two types of graphs.

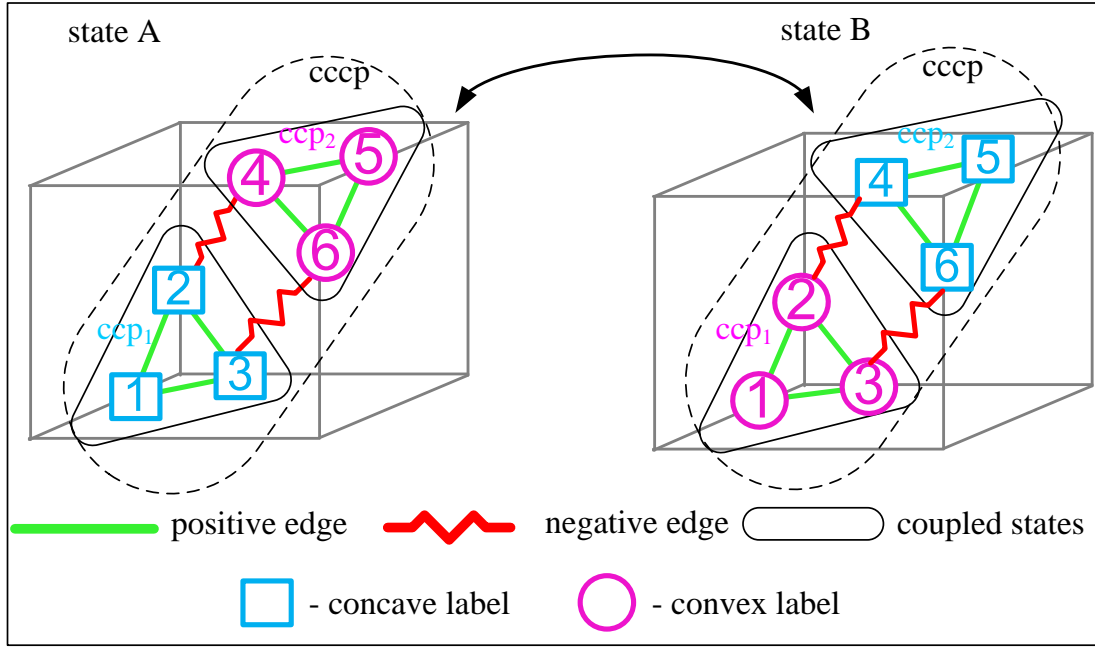


Figure 6.2: Swapping between the two interpretations of the Necker Cube. Locally coupled labels are swapped with alternate labelings to enforce global consistency. See text for explanation.

Adjacency graph where each node is an entity, such as a pixel, a superpixel, a line, or an object, which have to be labeled in K -classes (or colors). Most MRF and CRF used in computer vision are adjacency graphs.

Candidacy graph where each node is a candidate or hypothesis, such as a potential label for an entity, or a detected object instance in a window, which have to be confirmed ('on') or rejected ('off'). In other words, the graph is labeled with $K = 2$ colors.

As we will shown in Section 6.2.1, the adjacency graph can always be transferred to a bigger candidacy graph. In both cases, the tasks are posed as graph coloring problems on MRF, CRF or hierarchical graphs. There are two types of edges expressing either hard or soft constraints (or coupling) between the nodes.

Positive edge for a cooperative constraint that favors the two nodes having the same label in an adjacency graph or being turned on (or off) simultaneously in a candidacy graph.

Negative edge for a competitive or conflicting constraint that requires the two nodes to have different labels in an adjacency graph or one node to be turned on and the other turned off in a candidacy graph.

In Figure 6.2, the Necker cube is represented in an adjacency graph with each line being a node. The six internal lines are linked by 6 positive edges (in green) and two negative edges (in red and wiggly). Lines 2 and 4 have a negative edge between them as they intersect with each other, as do lines 3 and 6. We omit the labeling of the six outer lines for clarity.

In our formulation, the edges play computational roles, and are used to dynamically group nodes which are strongly coupled. On each positive or negative edge, we define an *edge probability* (using bottom-up discriminative models) for the coupling strength. Then we design a *protocol* for turning these edges on and off independently according to their edge probabilities respectively for each iteration. The protocol is common for all problems while the edge probabilities are problem specific. This probabilistic procedure turns off some edges, and all the edges that remain 'on' partition the graph into some connected components (*ccp*'s).

A *ccp* is a set of nodes that are connected by the positive edges. For example, Figure 6.2 has two *ccp*'s: ccp_1 includes nodes 1-2-3 and ccp_2 includes nodes 4-5-6. Each *ccp* is a locally coupled sub-solution.

A *cccp* is a composite connected component that consists of a number of *ccp*'s connected by negative nodes. For example, Figure 6.2 has one *cccp* containing ccp_1 and ccp_2 . Each *cccp* contains some conflicting sub-solutions.

At each iteration, C^4 selects a *cccp* and updates the labels of all nodes in the *cccp*

simultaneously so that (i) nodes in each *ccp* keep the same label to satisfy the positive or coupling constraints, and (ii) different *ccp*'s in the *cccp* are assigned different labels to observe the negative constraints.

Since C^4 can update a large number of nodes in a single step, it can move out of local modes and jump effectively between multiple solutions. The protocol design groups the *cccp*'s dynamically and guarantees that each step follows the MCMC requirements, such as detailed balance equations and thus it samples from the posterior probability.

We evaluate C^4 against other popular algorithms in the literature by two criteria.

1. The speed that they converge to solutions. In some studied cases, we know the global minimum solutions.
2. The number of unique solution states generated by the algorithms over time. This measures how “dynamic” an algorithm is.

The remainder of the chapter is organized as follows: In Section 6.2 we describe the graph representation and an overall protocol for C^4 . In Section 6.3 we introduce the C^4 algorithm on flat graphs and show the sampling of Potts models with positive and negative edges as special case. In Section 6.4, we show experiments on generalized C^4 outperforming BP, graph cuts, SW and ICM for some segmentation, labeling, and CRF inference tasks. We extend C^4 to hierarchical graphs in Section 6.5 and show experiments for hierarchical C^4 . Finally we conclude the chapter with a discussion of our findings in Section 6.6.

6.2 Graphs, Coupling and Clustering

6.2.1 Adjacency and Candidacy Graphs

We start with a flat graph G that we will extend to a hierarchical graph in Section 6.5,

$$G = \langle V, E \rangle, \quad E = E^+ \cup E^-. \quad (6.1)$$

Here $V = \{v_i, i = 1, 2, \dots, n\}$ is a set of vertices or nodes on which variables $X = (x_1, \dots, x_n)$ are defined, and $E = \{e_{ij} = (v_i, v_j)\}$ is a set of edges which is divided into E^+ and E^- for positive (cooperative) and negative (competitive or conflicting) constraints respectively. We consider two types of graphs for G .

Adjacency graph, where each node $v_i \in V$ is an entity, such as a pixel or super-pixel in image labeling, a line in a line drawing interpretation, or an object in scene understanding. Its variable $x_i \in \{1, 2, 3, \dots, K_i\}$ is a label or color. MRFs and CRFs in the literature belong to this category, and the task is to color the nodes V in K colors.

Candidacy graph, where each node $v_i \in V$ is a candidate or hypothesis, such as a potential label assignment for an entity, an object instance detected by bottom-up methods, or a potential match of a point to another point in graph matching. Its variable $x_i \in \{'on', 'off'\}$ is a boolean which confirms ('on') or rejects ('off') the candidate. In other words, the graph is labeled with $K = 2$ colors. In the graph matching literature (Chui and Rangarajan, 2003), the candidacy graph is represented by an assignment matrix.

An adjacency graph can always be transferred to a bigger candidacy graph by converting each node v_i into K_i nodes $\{x_{ij}\}$. $x_{ij} \in \{'on', 'off'\}$ represents $x_i = j$ in the adjacency graph. These nodes observe a mutual exclusion constraint to prevent fuzzy assignments to x_i .

Figure 6.3 shows this conversion. The adjacency graph $G_{adj} = \langle V_{adj}, E_{adj} \rangle$ has

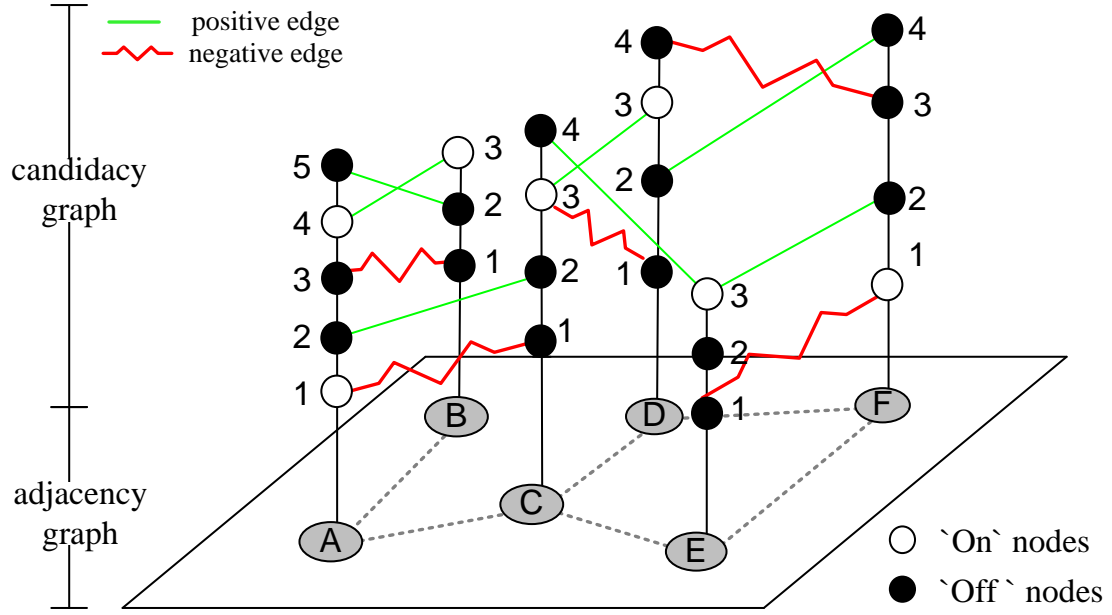


Figure 6.3: Converting an adjacency graph to a candidacy graph. The candidacy graph has positive (straight blue lines) and negative (wiggled red lines) edges depending on the values assigned to the nodes in the adjacency graph.

six nodes $V_{adj} = \{A, B, C, D, E, F\}$ and each has 3 \sim 5 potential labels. The variables are $X_{adj} = (x_A, \dots, x_F)$ with $x_A \in \{1, 2, 3, 4, 5\}$ and so on. We convert it to a candidacy graph $G_{can} = \langle V_{can}, E_{can} \rangle$ with 24 nodes $V_{can} = \{A_1, \dots, A_5, \dots, F_1, \dots, F_4\}$. Node A_1 represents a candidate hypothesis that assigns $x_A = 1$. The $X_{can} = (x_{A_1}, \dots, x_{F_4})$ are boolean variables.

Represented by the graph G , the vision task is posed as an optimization problem that computes a most probable interpretation with a posterior probability $p(X|I)$ or an energy function $\mathcal{E}(X)$.

$$X^* = \arg \max p(X|I) = \arg \min \mathcal{E}(X). \quad (6.2)$$

To preserve the ambiguity and uncertainty, we may compute multiple distinct solutions

$\{X_i\}$ with weights $\{\omega_i\}$ to represent the posterior probability.

$$(X_i, \omega_i) \sim p(X|I), \quad i = 1, 2, \dots, K. \quad (6.3)$$

6.2.2 Positive and Negative Edges

In conventional vision formulation, edges in the graphs are a representational concept and the energy terms in \mathcal{E} are defined on the edges to express the interactions between nodes. In contrast, Swendsen-Wang (Swendsen and Wang, 1987) and Edward-Sokal (Edwards and Sokal, 1988) added a new computational role to the edges in their cluster sampling method. The edges are turned ‘on’ and ‘off’ probabilistically to dynamically form groups (or clusters) of nodes which are strongly coupled. We will introduce the clustering procedure shortly after the example below. We adopt this notion and the edges in graph G are characterized in three aspects:

Positive vs negative. A positive edge represents a cooperative constraint for two nodes having the same label in an adjacency graph or being turned on (or off) simultaneously in a candidacy graph. A negative edge requires the two nodes to have different labels in an adjacency graph or requires one node to be turned on and the other turned off in a candidacy graph.

Hard vs soft. Some edges represent hard constraints which must be satisfied, for example, in line drawing interpretation or scene labeling, while other edge constraints are soft and can be expressed with a probability.

Position dependent vs value dependent. Edges in adjacency graphs are generally *position dependent*. For example, in an Ising model an edge between two adjacent nodes poses a soft constraint that they should have the same label (ferromagnetism) or opposite labels (antiferromagnetism). In contrast, edges in candidacy graphs are *value dependent* and thus have more expressive power. This is common for vision tasks,

such as scene labeling, line drawing interpretation, and graph matching. As Figure 6.3 illustrates, the edges between nodes in the candidacy graph could be either positive or negative depending on the values assigned to nodes A, B in the adjacency graph.

As we will show in a later subsection that the positive and negative edges are crucial for generating connected components and resolving the problem of node coupling.

6.2.3 The Necker Cube Example

Figure 6.4 shows the construction of a candidacy graph G for interpreting the Necker cube. For clarity of discussion we assume the exterior lines are labeled and the task is to assign two labels (concave and convex) to the six inner lines such that all local and global constraints are satisfied. Therefore we have a total of 12 candidate assignments or nodes in G .

Based on the theory of line drawing interpretation (Sugihara, 1986; Mackworth, 1973), the two 'Y'-junctions pose positive constraints so that lines 1-2-3 have the same label and lines 4-5-6 have the same label. We have 12 positive edges (green) in G to express these constraints. The intersection of lines 2 and 4 poses negative constraints that lines 2 and 4 have opposite labels which are shown in the red and wiggly edges in Figure 6.4. The same is true for lines 3 and 6. The two different assignments for each line should also be linked by a negative edge. These negative edges are not shown for clarity.

What does a solution to a graph G look like? We can see in Figure 6.4 what the two Necker Cube solutions would look like. The first would have all nodes 1,2, and 3 labeled convex and all nodes 4,5, and 6 labeled concave. In this case, all edge constraints are satisfied and no underlying graph nodes have more than one value assigned to them. This would create a valid 3D interpretation where the cube is “coming out” of the page. The alternative solution has the opposite labeling, and creates a 3D in-

candidacy graph

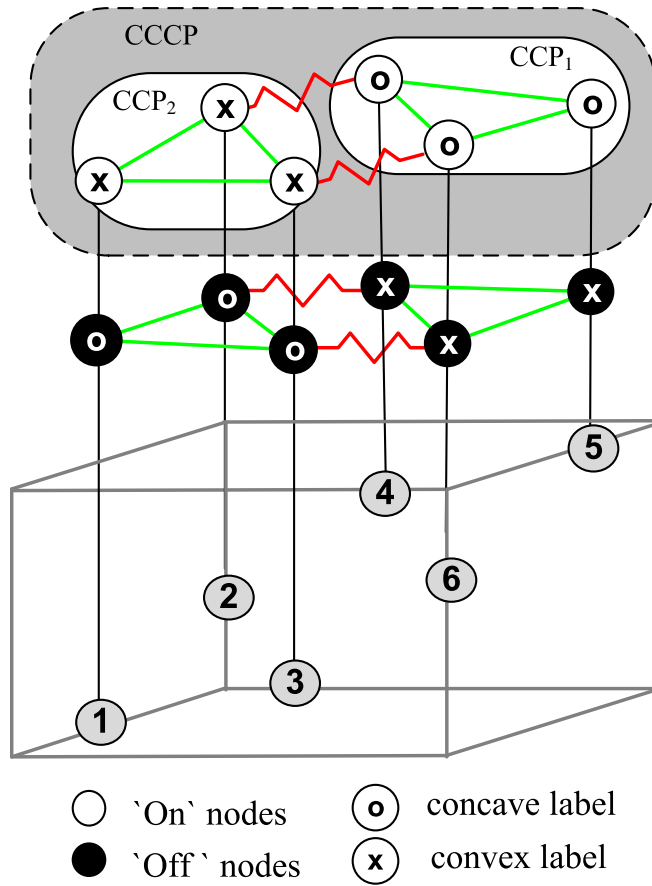


Figure 6.4: The Necker cube example. The adjacency graph with 6 nodes (bottom) is converted to a candidacy graph of 12 nodes (top) for concave and convex label assignments respectively. 12 positive and 2 negative edges are placed to ensure consistency.

terpretation of the cube “going in” to the page. No other assignments of labels would succeed in satisfying all the constraints.

Let us now look at what it would take to switch from one solution to the other. Each set of nodes, 1-2-3 and 4-5-6, constitutes a corner of the Necker Cube and all have positive constraints between them. This indicates that we should update all of these values simultaneously. We can create a *connected component*, or *ccp*, of these nodes, which consists of all nodes currently satisfying their positive constraints. This results in ccp_1 and ccp_2 , comprised of nodes 1-2-3 and nodes 4-5-6 respectively.

If we were simply to invert the labels of ccp_1 , we would swap all labels from concave to convex or vice versa. Thus the ccp has captured that all of its nodes should change at once, and creates an effective swap of the corner label. However, simply swapping ccp_1 alone would create an inconsistent interpretation, as all edges in the whole graph would now have the same label. What we need to do is simultaneously swap ccp_1 and ccp_2 , thus switching both corners at once.

Notice that we have negative edges between nodes 2 and 4 and between nodes 3 and 6. Negative edges can be thought of as indicators of multiple competing solutions, as they necessarily dictate that groups on either end of the edge can either be (‘on’, ‘off’) or (‘off’, ‘on’), creating two possible outcomes. This negative edge connects nodes in ccp_1 and ccp_2 , thus indicating that those nodes in the two ccps must have different labels. Because we know that all nodes within ccp_1 and ccp_2 must have the same label, this negative edge effectively means ccp_1 and ccp_2 must have opposite labels. If we construct a composite connected component (called *cccp*), $cccp_{12}$, encompassing nodes 1-6, we now have a full component that contains all relevant constraints. Moving from solution 1 to 2 is now as simple as flipping all the nodes simultaneously, or equivalently satisfying all of the constraints.

In the next subsection, we explain how we form the ccp’s and cccp’s in a formal

way.

6.2.4 Edge Probability for Clustering

On each positive or negative edge, we define an *edge probability* (using bottom-up discriminative models) for the coupling strength. That is, at each edge $e \in E$, we define an auxiliary probability $u_e \in \{0, 1\}$ or $\{'on', 'off'\}$, which follows an independent probability q_e .

In Swendsen and Wang (Swendsen and Wang, 1987), the definition of q_e is decided by the energy term in the Potts model $q_e = e^{-2\beta}$ as a constant for all e . Barbu and Zhu (Barbu and Zhu, 2005), for the first time, separate q_e from the energy function and define it as a bottom-up probability: $q_e = p(l(x_i) = l(x_j) | F(x_i), F(x_j)) = p(e = on | F(x_i), F(x_j))$ with $F(x_i)$ and $F(x_j)$ being local features extracted at node x_i and x_j . This can be learned through discriminative training, for example, by logistic regression and boosting,

$$\frac{p(l(x_i) = l(x_j) | F(x_i), F(x_j))}{p(l(x_i) \neq l(x_j) | F(x_i), F(x_j))} = \sum_n \lambda_n h_n(F(x_i), F(x_j)).$$

On a positive edge $e = (i, j) \in E^+$, $u_e = 'on'$ follows a Bernoulli probability,

$$u_e \sim \text{Bern}(q_e \cdot 1(x_i = x_j)).$$

$1(\cdot)$ is boolean function. It equals 1 if the condition is satisfied and 0 otherwise. Therefore, at the present state X , if the two nodes have the same color, i.e. $x_i = x_j$, then the edge e is turned on with probability q_e . If $x_i \neq x_j$, then $u_e \sim \text{Bern}(0)$ and e is turned off with probability 1. So, if two nodes are strongly coupled, q_e should have a higher value to ensure that they have a higher probability to stay the same color.

Similarly, for negative edges $e \in E^-$, $u_e = 'on'$ also follows a Bernoulli probability,

$$u_e \sim \text{Bern}(q_e 1(x_i \neq x_j)).$$

At the present state X , if the two nodes have the same color $x_i = x_j$, then the edge e is turned off with probability 1, otherwise e is turned on with probability q_e to enforce that x_i and x_j stay in different colors.

After sampling u_e for all $e \in E$ independently, we denote the sets of positive and negative edges that remain 'on' as $E_{on}^+ \subset E_+$ and $E_{on}^- \subset E_-$ respectively. Then we have a formal definitions of the *ccp* and *cccp*.

Definition 1 A *ccp* is a set of vertices $\{v_i; i = 1, 2, \dots, k\}$ for which every vertex is reachable from every other vertex by the positive edges in E_{on}^+ .

Definition 2 A *cccp* is a set of *ccps* $\{ccp_i; i = 1, 2, \dots, m\}$ for which every *ccp* is reachable from every other *ccp* by the negative edges in E_{on}^- .

No two *ccp*'s are reachable by positive edges, or else they would be a single *ccp*. Thus a *cccp* is a set of isolated *ccp*'s that are connected by negative edges. An isolated *ccp* is also treated as a *cccp*.

In Section 6.5, we will treat the invalid cases where a *ccp* contains negative edges by converting it to a *cccp*.

To observe the detailed balance equations in MCMC design, we need to calculate the probabilities for selecting a *ccp* or *cccp* which are determined by the edge probabilities q_e . For this purpose we define their cuts. In general, a cut is the set of all edges connecting nodes between two nodes sets.

Definition 3 Under a current state X , a cut for a *ccp* is the set all positive edges between nodes in *ccp* and its surrounding nodes which have the same label,

$$Cut(ccp|X) = \{e : e \in E^+, x_i = x_j, i \in ccp, j \notin ccp\}.$$

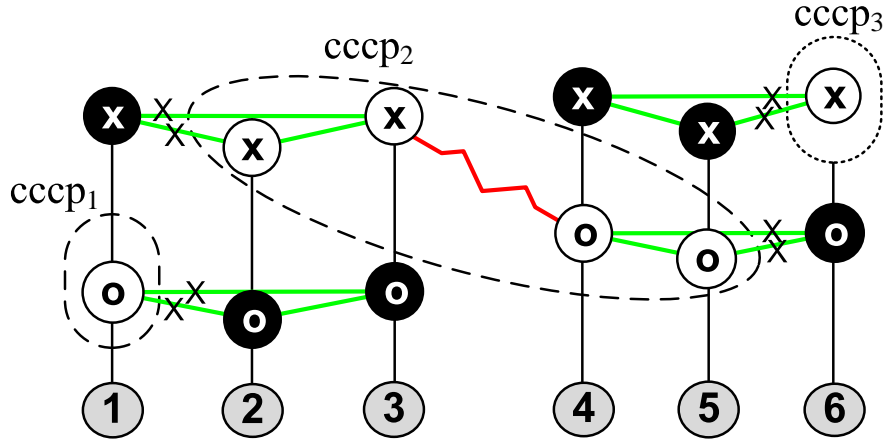


Figure 6.5: A Necker cube candidacy graph not in a solution state.

These are the edges that must be turned off probabilistically (with probability $1 - q_e$) in order to form the ccp and the cut depends on the state X .

Definition 4 A cut for a $cccp$ at a state X is the set of all negative (or positive) edges connecting the nodes in the $cccp$ and its neighboring node which have different (or same) labels,

$$\begin{aligned} Cut(cccp|X) = & \{e : e \in E^-, i \in cccp, j \notin cccp, x_i \neq x_j\} \\ & \cup \{e : e \in E^+, i \in cccp, j \notin cccp, x_i = x_j\}. \end{aligned}$$

All these edges must be turned off probabilistically with probability $1 - q_e$ in order to form the composite connected component $cccp$ at state X .

As edges in E_{on}^+ only connect nodes with the same label, so all nodes in a ccp have the same label. In contrast, all edges in E_{on}^- only connect nodes with different labels, adjacent ccp 's in a $cccp$ must have different labels.

To illustrate the concepts, we show a non-solution state X for the Necker cube in Figure 6.5. By turning off some edges (marked with the crosses), we obtain three

$cccp$'s for the nodes that are currently 'on'. In this example, $q_e = 1$, as these are hard constraints that are inviolable. $cccp_1$ and $cccp_3$ have only 1 node, and $cccp_2$ has two ccp 's with 4 nodes. The algorithm will now arbitrarily select a $cccp$ and update its values according to its constraints. If it selects either $cccp_1$ or $cccp_3$, then we are one step closer to the solution. If it selects ($cccp_2$), then all the 4 vertex labels are swapped and we have reached a solution state and will continue to swap back and forth between the two solutions.

6.3 C^4 algorithm on flat graphs

In this section, we introduce the C^4 algorithm for cluster sampling on flat graphs.

6.3.1 Outline of the algorithm

The C^4 algorithm works iteratively following the MCMC design. In each iteration, it generates the $cccp$'s, selects (or visits) a $cccp_o$ with a probability, and reassigns labels to its ccp 's such that all internal negative constraints are satisfied. As the number of ccp 's in $cccp_o$ grows large, the number of potential labelings will grow as well. One can remedy this situation in two ways:

1. Use a constraint-satisfaction problem (CSP)-solver to solve this smaller, easier constraint satisfaction problem within $cccp_o$.
2. Use random or heuristic sampling to find a new valid labeling.

We will use the second approach throughout this dissertation and the number of ccp 's in a $cccp_o$ is in general small, so the label assignment is not a problem. The C^4 algorithm can be viewed as a method that breaks a large constraint-satisfaction problem into smaller fragments in $cccp_o$ which can be satisfied locally. Then it propa-

gates the solution through iterations.

This assignment represents a move in MCMC which is accepted by the Metropolis-Hastings step with an acceptance probability. The acceptance probability account for the probabilities for generating the $cccp$'s, selecting a $cccp_o$, assigning new labels, and the posterior probability.

In summary, we state the C^4 algorithm below.

C^4 algorithm

Input: A graph $G = \langle V, E \rangle$ and posterior prob. $p(X|I)$.

Calculate the edge probability $q_e, \forall e \in E$.

// q_e is a problem specific discriminative probability.

Initialize the state $X = (x_1, x_2, \dots, x_n)$.

// e.g. all nodes are turned off in a candidacy graph.

Repeat

Denote the current X by state A .

Step 1: generating a $cccp_o$ at state A

$\forall e = (i, j) \in E^+$, sampling $u_e \sim \text{Bern}(q_e 1(x_i = x_j))$

$\forall e = (i, j) \in E^-$, sampling $u_e \sim \text{Bern}(q_e 1(x_i \neq x_j))$

Generating the $\{ccp\}$ and $\{cccp\}$ based on E_{on}^+ and E_{on}^-

Selecting a $cccp_o$ from $\{cccp\}$ probabilistically

// Denote the prob for selecting $cccp_o$ by $q(cccp_o|A)$.

Step 2: Assigning labels to ccp 's in the $cccp$ with

probability: $q(l(cccp_o = L|cccp_o, A))$.

Denote the new X as state B .

Step 3: Calculating the acceptance probability:

$$\alpha(A \rightarrow B) = \min(1, \frac{q(B \rightarrow A)}{q(A \rightarrow B)} \cdot \frac{p(X=B|\mathbf{I})}{p(X=A|\mathbf{I})}).$$

Output: distinct states $\{X^*\}$ with highest probabilities.

We will elaborate on the probabilities used in the algorithm in the next subsection,

Intuitively, C^4 is a general procedure for designing a cluster sampling algorithm, just like SVM is a general design for classifiers. For specific problems, one needs to select the probabilities for q_e and $q(l(cccp_o) = L|cccp_o, A)$. But unlike SVM which needs good features to work, these probabilities are not very critical to its convergence.

6.3.2 Calculating the Acceptance Probability

In Markov chain design, each move between two states A and B is made reversible and observes the detailed balance equation,

$$p(X = A|I)K(A \rightarrow B) = p(X = B|I)K(B \rightarrow A). \quad (6.4)$$

$K(A \rightarrow B)$ is the Markov chain kernel or transition probability from A to B . In the Metropolis-Hastings design,

$$K(A \rightarrow B) = q(A \rightarrow B)\alpha(A \rightarrow B), \quad \forall A \neq B. \quad (6.5)$$

$q(A \rightarrow B)$ is the probability for proposing state B from state A , and $\alpha(A \rightarrow B)$ is the acceptance probability,

$$\alpha(A \rightarrow B) = \min(1, \frac{q(B \rightarrow A)}{q(A \rightarrow B)} \cdot \frac{p(X = B|\mathbf{I})}{p(X = A|\mathbf{I})}). \quad (6.6)$$

It is easy to check that the design of proposal probability in eqn.(6.6) and the acceptance probability in eqn.(6.5) makes the kernel satisfy the detailed balance equation in (6.4), which in turn suffices to observe the invariance condition,

$$p(X = A|I)K(A \rightarrow B) = p(X = B|I). \quad (6.7)$$

So, $p(X|I)$ is the invariant probability of the Markov chain with kernel K . Now we elaborate on the design of proposal and acceptance probabilities. The acceptance probability is determined by two ratios.

(i) The ratio $\frac{p(X=B|I)}{p(X=A|I)}$ is problem specific and is not part of our design. The posterior probability can be in general form and does not have to be modified or approximated to fit the C^4 algorithm. As states A and B only differ in their labels for nodes in $cccp_o$, it often can be computed locally if the posterior probability is a MRF or CRF.

(ii) The proposal probability ratio is completely up to our design, and it includes two parts,

$$\frac{q(B \rightarrow A)}{q(A \rightarrow B)} = \frac{q(cccp_o|B)}{q(cccp_o|A)} \cdot \frac{q(l(cccp_o) = L_A|cccp_o, B)}{q(l(cccp_o) = L_B|cccp_o, A)}.$$

$q(cccp_o|A)$ and $q(cccp_o|B)$ are the probabilities for choosing $cccp_o$ at states A and B respectively. Given the chosen composite connected component $cccp_o$, in both states A and B , the assignment of new labels is independent of the surrounding neighbors of $cccp_o$ and is often assigned by equal probability (uniform) among all valid assignments in the CSP-solver. Thus they cancel out, and we have $\frac{q(l(cccp_o)=L_A|cccp_o, B)}{q(l(cccp_o)=L_B|cccp_o, A)} = 1$.

To summarize, the key to the algorithm design is the ratio $\frac{q(cccp_o|B)}{q(cccp_o|A)}$. In single site sampling, such as Gibbs sampler, each node is a $cccp_o$ and the selection is simply a visiting scheme. In C^4 , the probability for choosing $cccp_o$ at a state depends on two steps: (a) How likely it is to generate $cccp_o$ by sampling the edge probabilities q_e following the Bernoulli probability. (b) How likely it is to select $cccp_o$ from the set of formed $\{cccp\}$ in states A and B . These probabilities are hard to compute, because there are a vast amount of partitions of the graph that include a certain $cccp_o$ by turning on/off edges. A partition is a set of $cccp$'s after turning off some edges.

Interestingly, the set of all possible partitions in state A is identical to those in state B , and all these partitions must share the same cut $Cut(cccp_o)$. That is, in order for $cccp_o$ to be a composite connected component, its connections with its neighboring

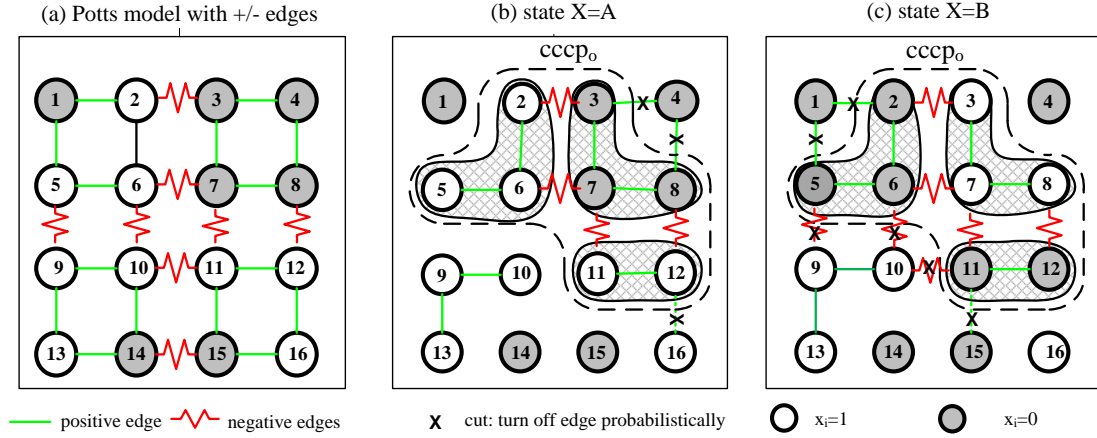


Figure 6.6: The Potts model with negative edges. (a) Minimum energy is a checker-board pattern. (b) Forming $cccp_o$. (c) $cccp_o$ consists of sub-ccps of positive edges connected by negative edges.

nodes must be turned off. Even though the probabilities are in complex form, their ratio is simple and clean due to cancellation. Furthermore, given the partition, $cccp_o$ is selected with uniform probability from all possible $cccp$'s.

Proposition 1 *The proposal probability ratio for selecting $cccp_o$ at states A and B is*

$$\frac{q(cccp_o|B)}{q(cccp_o|A)} = \frac{\prod_{e \in Cut(cccp_o|B)} (1 - q_e)}{\prod_{e \in Cut(cccp_o|A)} (1 - q_e)}. \quad (6.8)$$

We will prove this in the Appendix C in a similar way to the SW-cut method (Barbu and Zhu, 2005)

6.3.3 Special case: Potts model with +/- edges

To illustrate C^4 , we derive it in more detail for a Potts model with positive and negative edges. Let X be a random field defined on a 2D lattice with discrete states

$x_i \in \{0, 1, 2, \dots, L - 1\}$. Its probability is specified by

$$p(X) = \frac{1}{Z} \exp\{-\mathcal{E}(X)\}; \quad (6.9)$$

$$\mathcal{E}(X) = \sum_{\langle i, j \rangle \in E^+} \beta \delta(x_i = x_j) + \sum_{\langle i, j \rangle \in E^-} \beta \delta(x_i \neq x_j),$$

where $\beta > 0$ is a constant. The edge probability will be $q_e = 1 - e^{-\beta}$ for all edges.

Figure 6.6(a) shows an example on a small lattice with $L = 2$ labels, which is an adjacency graph with position dependent edges. The states with checkerboard patterns will have highest probabilities. Figure 6.6(b) and (c) show two reversible states A and B by flipping the label of a $cccp_o$ in one step. In this example, $cccp_o$ has three ccp 's, $cccp_o = \{\{2, 5, 6\}; \{3, 7, 8\}; \{11, 12\}\}$. The labels of the 8 nodes are re-assigned with uniform probability, and this leads to the difference in the cuts for $cccp_o$ at the two states, $Cut(cccp_o|A) = \{(3, 4), (4, 8), (12, 16)\}$ and $Cut(cccp_o|B) = \{(1, 2), (1, 5), (5, 9), (6, 10), (10, 11), (11, 15)\}$.

Proposition 2 *The acceptance probability for C^4 on the Potts model is $\alpha(A \rightarrow B) = 1$ for any two states with different labels in $cccp_o$. Therefore, the move is always accepted.*

The proof follows two observations. Firstly, the energy terms inside and outside $cccp_o$ are the same for both A and B , and they differ only at the cuts of $cccp_o$. More precisely, let $c = |Cut(cccp_o|B)| - |Cut(cccp_o|A)|$ be the difference of sizes in the two cuts (i.e. $c = 3$ in our example), it is not too hard to show that

$$\frac{p(X = B | \mathbf{I})}{p(X = A | I)} = e^{-\beta c} \quad (6.10)$$

Secondly, we have the proposal probability ratio, following eqn.(6.8),

$$\frac{q(cccp_o|B)}{q(cccp_o|A)} = \frac{(1 - q_e)^{|Cut(cccp_o|B)|}}{(1 - q_e)^{|Cut(cccp_o|A)|}} = e^{\beta c}. \quad (6.11)$$

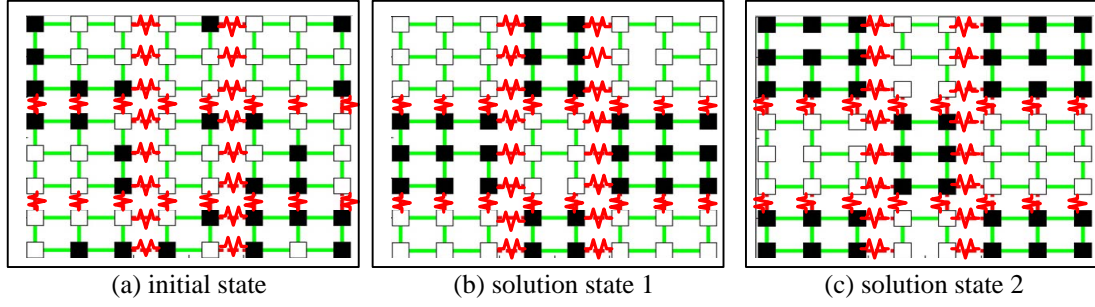


Figure 6.7: The checkerboard Ising/Potts model solutions.

Plugging the two ratios in eqn.6.6, we have $\alpha(A \rightarrow B) = 1$. In the literature of SW (Edwards and Sokal, 1988), Edwards and Sokal explain the SW on Potts model as data augmentation where the edge variables $\{u_e\}$ are treated as auxiliary variables and they sample $\{x_i\}$ and $\{u_e\}$ iteratively from a joint probability.

6.4 Experiments on Flat Graphs

In this section we test C^4 's performance on some flat graphs (MRF and CRF) in comparison with the Gibbs sampler (Geman and Geman, 1984), SW method (Swendsen and Wang, 1987), iterated conditional modes (ICM), graph cuts (Boykov et al., 2001), and loopy belief propagation (LBP) (Kumar and Torr, 2006). We choose classical examples: (i) Ising/Potts model for MRF; (ii) Line drawing interpretation for constrained-satisfaction problem using candidacy graph and (iii) scene labeling using CRF. This section serves as an experimental study of C^4 in general, while we will apply C^4 to our object recognition and aerial modeling problems in Chapter 7

6.4.1 Checkerboard Ising Model

We first show the Ising model on a 9×9 lattice with positive and negative edges (the Ising model is a special case of the Potts model with $L = 2$ labels). We tested

C^4 with two parameters settings: (i) $\beta = 1$ and thus $q_e = 0.632$; and (ii) $\beta = 5$ and thus $q_e = 0.993$. In this lattice we've created a checkerboard pattern. We've assigned negative and positive edges so that blocks of nodes want to be the same color, but these blocks want to be different colors than their neighbors.

Figure 6.7 shows a typical initial state to start the algorithm, and two perfect solutions with minimum (i.e. 0) energy. To get a better idea of C^4 's performance in finding the two solutions, Figure 6.8(a) shows a plot of energy versus time for C^4 , Gibbs sampler, SW, graph cuts, and LBP. C^4 converges second fastest of all five algorithms in a mere 10 iterations or so, behind graph cuts. Belief propagation cannot converge due to the loopiness of the graph, and Gibbs sampler and the conventional Swendsen-Wang cannot quickly satisfy the constraints as they do not update enough of the space at each iteration. This shows that C^4 has a very low burn-in time.

To get a better idea of where C^4 really excels, we turn to Figure 6.8(b) and (c). Figure 6.8(b) shows the state visited in at each iteration. We show the states in 3 levels: the curve hits the ceiling or floor for the two minimum energy states respectively, and the middle for all other states. Here we are only comparing graph cuts, SW and C^4 as they are the only algorithms that converge to a solution in a reasonable amount of time. What is impressive to note is that C^4 keeps swapping solutions while SW and graph cuts get stuck in their first solution. This is because C^4 can group along negative edges as well as positive edges to update large portions of the system at once, while Swendsen-Wang is stuck proposing low probability moves over smaller portions of the solution space.

We also compared our results for experiments where $\beta = 1$ and $\beta = 5$. Figure 6.8(c) shows the states visited by the sampler over time. In the $\beta = 1$ case, it clearly takes longer for C^4 to converge, because it can't form large components with high probability. Also, once it does, it gets stuck in the first solution because the edge prob-

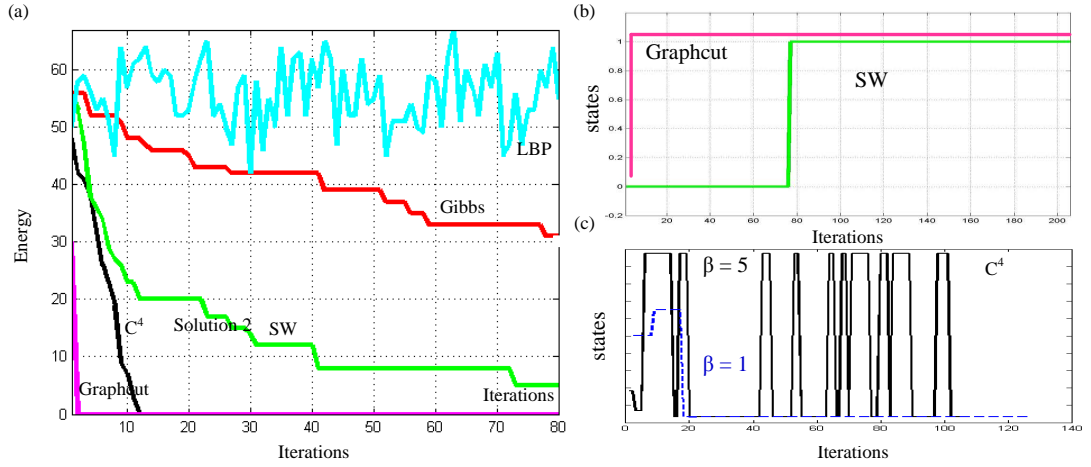


Figure 6.8: (a) Energy plots of C^4 , SW, Gibbs sampler, graph cuts, and LBP on the Ising model vs. time. (b) (c) The state (visited by the algorithms) in time for graph cuts, SW and C^4 . Once SW and graph cuts hit the first solution, they get stuck while C^4 keeps swapping between the two minimum energy states. C^4 results shown for $\beta = 1$ and $\beta = 5$.

abilities are so low that they don't form a large enough component to switch out. As β gets large, however, C^4 very quickly takes steps in the space towards the solution and can move rapidly between solution states. We have found that an annealing schedule where $q_e = 1 - e^{-\beta/T}$ and T is adjusted such that q_e moves from 0 to 1 over the course of the experiment works quite well too.

6.4.2 Checkerboard Potts Model with 7 Labels

We ran the same experiment as with the Ising model above but this time solved the same checkerboard pattern on a Potts model in which each site could take one of seven possible colors ($L = 7$). In this example, we have a large number of equal states (in checkerboard pattern) with minimum energy.

Figure 6.9(a) plots the energy convergence of each algorithm over time. Graph

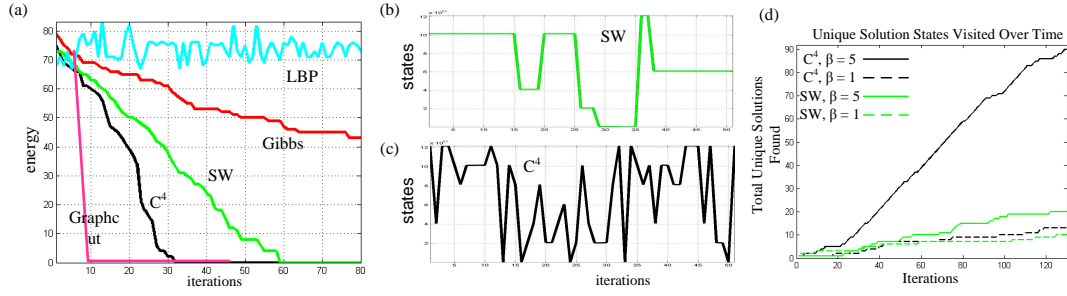


Figure 6.9: (a) Energy plots of C^4 , SW, Gibbs sampler, and LBP on the Potts model ($L = 7$) vs. time. (b) (c) The minimum energy states visited by SW and C^4 algorithms over time. (d) Total number of unique solutions found vs. time for SW and C^4 with $\beta = 1$ and $\beta = 5$.

cuts again converges to just one of the many solutions. Unlike in the case of the $L = 2$ model, SW is able to find multiple solutions this time, as seen in Figure 6.9(b). This is because SW can now update a block of the checkerboard with less constraints in label assignments. Figure 6.9(c) shows the number of distinct states with minimum energy that have been visited by SW and C^4 over time. We see that C^4 explores more states in a given time limit which again demonstrates that C^4 is more dynamic and thus has a fast mixing time – a crucial measure for the efficiency of MCMC algorithms. We also compare the case where $\beta = 1$ vs. $\beta = 5$. Once again, we see that $\beta = 1$ doesn't create strong enough connections for C^4 to move out of local minimum, so it finds roughly as many unique solutions as Swendsen-Wang does (about 13). When β is increased to 5, however, we see that the number of unique solutions C^4 finds in the same amount of time skyrockets from 13 to 90. We thus see that C^4 can move around the solution space much more rapidly than other methods when β is high and can discover a huge number of unique solution states.

6.4.3 Line Drawing Interpretation

The previous two examples are based on MRF models whose edges are position dependent. Now we test on line drawing interpretation on candidacy graph. We use two classical examples which have multiple stable interpretations, or solutions: (i) the Necker cube in Figure 6.1 that has two interpretations; and (ii) a line drawing with double cubes in Figure 6.10 that has four interpretations. The swapping between these states involves the flipping of 3 or 12 lines simultaneously. Our goal is to test whether the algorithms can compute the multiple distinct solutions over time.

We adopt a Potts like model on the candidacy graph. Each line in the line drawing is a node in the Potts model, which can take one of eight line drawing labels indicating whether the edge is concave, convex, or a depth boundary. See (Sugihara, 1986) for an in-depth discussion on labels for consistent line drawings. We add an edge in our candidacy graph between any two lines that share a junction. At each junction, there are only a small set of valid labels for each line that are realizable in a 3D world. We add positive edges between pairs of line labels that are consistent with one of these junction types, and negative edges between line labels that are not. Thus, we model the pairwise compatibility of neighboring line labels given the type of junction they form.

As an example, in the Necker cube, the bottom right corner of the cube is an “arrow” junction. Based on consistent line labeling, the outer edges of the arrow junction must both be convex, both be concave, or both be depth boundaries. Thus we add positive edges between these possibilities, and negative edges between all other possibilities. By enforcing these pairwise constraints, the candidacy graph implicitly represents the higher-level junction constraints. A valid, zero energy solution to the line drawing labeling will result in junctions that are all realizable in a 3D world, as defined in (Sugihara, 1986). For these experiments we set $\beta = 2$, resulting in $q_e = 0.865$.

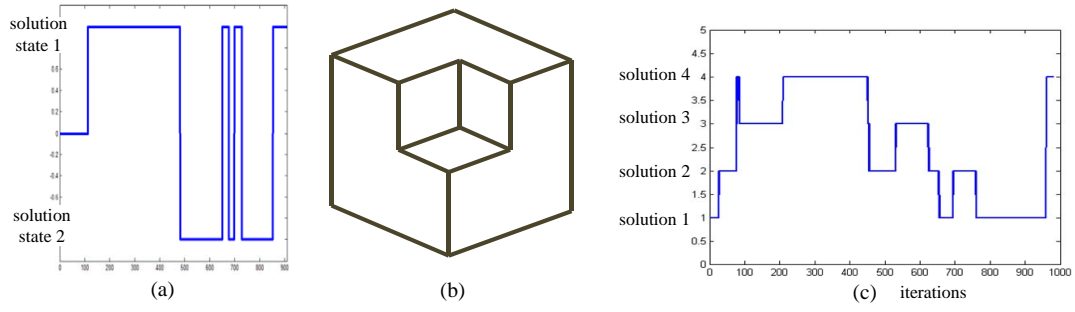


Figure 6.10: Experimental results for swapping state between interpretations: (a) States visited by C^4 for the Necker Cube. (b) A line drawing with outer and inner cubes. (c) States visited by C^4 for the double cubes.

Figures 6.10.(a) and (c) plot the state visited by the algorithms over time. Once again we see that C^4 can quickly switch between solutions where CSP solvers or other MCMC methods could get stuck.

6.4.4 Labeling Man-made Structures on CRFs

In (Kumar and Hebert, 2003), CRFs were learned to model man-made structures in images. Images of outdoor scenes, like those in Figure 6.11, are broken into 16×24 grids and each cell is assigned a label $x_i = \{-1, +1\}$ indicating if they covered man-made structure or not. The probability of labeling the sites x given data y in a Conditional random field (CRF) is

$$p(X|Y) = \frac{1}{Z} \exp \sum_i \phi(x_i, y) + \sum_i \sum_{j \in \mathcal{N}_i} \psi(x_i, x_j, y). \quad (6.12)$$

In other words, $p(X)$ is an MRF globally conditioned on the image Y . For space we refer the reader to (Kumar and Hebert, 2003) for more details of this model. We simply choose their model so that we can compare various algorithms on the same representation. The authors use a greedy algorithm (ICM) for inference.

We learned the CRF weights via BFGS (Fletcher, 1970) using the data from (Ku-

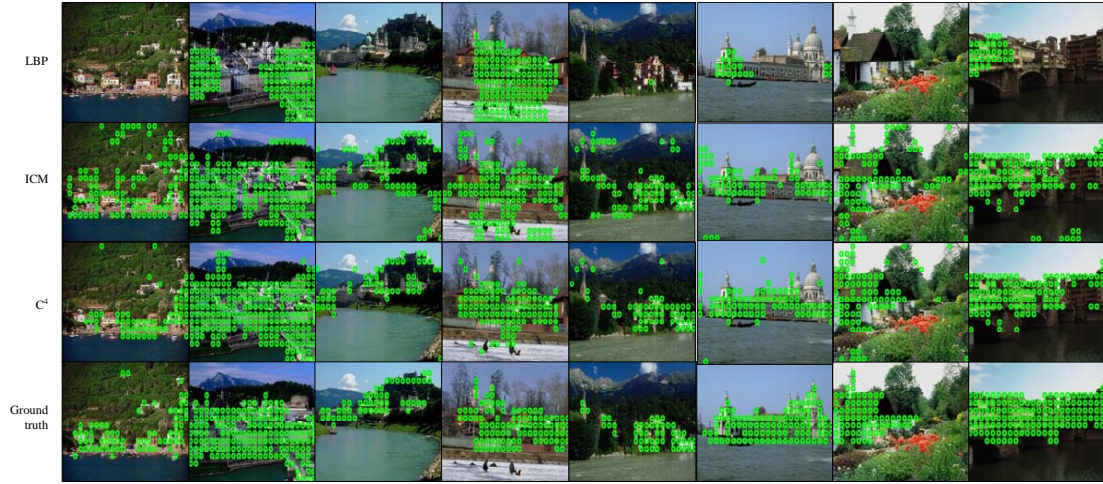


Figure 6.11: Man-made structure detection results.

mar and Hebert, 2003) and compared inference results using C^4 to ICM, LBP, and SW. Edge probabilities were taken from the CRF interaction potentials. The CRF defines a potential for the case when two sites have the same label and for when they have different labels. If the ratio of these two potentials was below a threshold τ , a negative edge was used to connect the sites to enforce them to be labeled differently. Such negative edges indicate strong boundaries of the label map. Note that the positive and negative edges are adopted for computational purpose here for generating their *cccp*'s and they do not alter the energy function or probability model.

Figure 6.11 shows the detection results and ground truths. LBP has very few false positives, but misses huge amounts of the detection. ICM looks graphically similar to C^4 , but produces significantly more false positives. C^4 is able to swap between foreground/background fragments in large steps so it can find blocks of man-made structure more effectively.

Table 6.1 shows our results as in (Kumar and Hebert, 2003). The reported false positive rate is much lower for ICM in (Kumar and Hebert, 2003). We were not able to find a setting for the CRF weights that recreated those results. This is not to say

Method	FalsePositive (per image)	DetectRate (%)
LBP	23.18	0.451
SW	156.05	0.468
ICM	61.78	0.697
C^4	47.12	0.696

Table 6.1: False positives per image and detection rate using Loopy BP, SW, ICM, and C^4 for man-made structure detection.

that (Kumar and Hebert, 2003) can't achieve those results, merely that, given the best weightings we could find using BFGS, ICM performed less well than C^4 did. This was the case in every trial we performed, though we could not recreate the low false positive rate in (Kumar and Hebert, 2003). For lack of a better form of comparison, we report the false positive rate using our implementation along with our false positive rate using C^4 .

6.5 C^4 on Hierarchical Graphs

In this section, we discuss the consistency of the flat graphs and extend C^4 from flat graphs to hierarchical graphs and then we address high-order constraints that involve more than two sites.

6.5.1 Condition for Graph Consistency

In each iteration of the C^4 algorithm, suppose we have turned on edges probabilistically and the original graph $G = \langle V, E \rangle$ becomes $G_{on} = \langle V, E_{on} \rangle$ with $E = E_{on} \cup E_{off}$, $E_{on} = E_{on}^+ \cup E_{on}^-$, and $E_{off} = E_{off}^+ \cup E_{off}^-$. As we discussed in Section 6.2.4 all nodes in the graph G_{on} in each *ccp* shares the same label and they

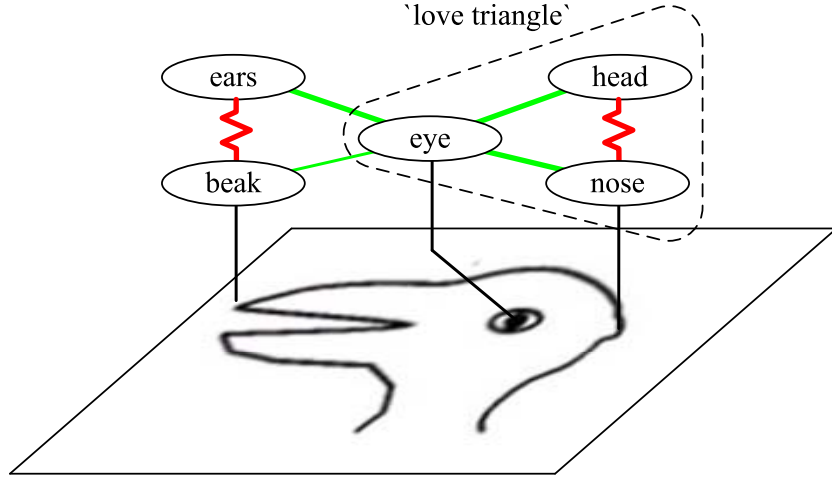


Figure 6.12: An attempt to solve the duck/rabbit illusion using flat C^4 . We see that we are very likely to form love triangles on the left and right of the graph, making constraint satisfaction very difficult.

are supposed to form a coupled partial solution. However, if the constraints in graph G is inconsistent, then some nodes in a ccp may be connected by edges in E_{off}^- . Though such negative edges are not turned on in ccp , They indicate that some nodes in the ccp may be conflicting to each other. This may not be a serious problem, for example, the negative edges may simply express soft constraints, such as overlapping windows due to occlusion, which is acceptable in the final solution.

Figure 6.12 shows an example where the negative edge is a hard constraint. If we try to solve the duck/rabbit illusion using flat candidacy graph, a ccp may contain $\{ 'eye', 'nose', 'head' \}$ which is inconsistent. We call it a "love triangle".

Definition 5 In a graph G , two nodes i, j connected by a negative edge is said to be involved in a love triangle if there also exists a path between i, j that consists of all positive edges.

Definition 6 A ccp is said to be consistent in graph G if there is no negative edges in

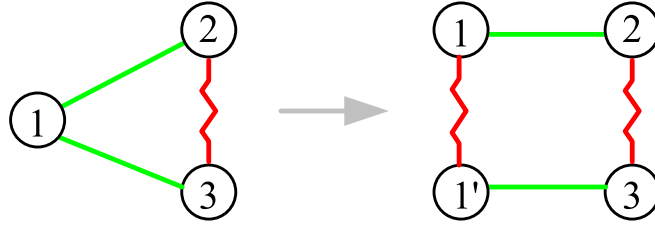


Figure 6.13: Breaking the 'love triangle' in a candidacy graph.

E that connect two nodes in the ccp , that is, $\{e : i, j \in ccp\} \cap E^- = \emptyset$. A graph G is said to be consistent if all its ccp 's are always consistent in C^4 .

When a graph is consistent, then we are guaranteed to get valid solutions.

The existence of the so-called 'love triangles' are the sole reason to generate inconsistent ccp 's. For this we can easily prove the following proposition.

Proposition 3 *In the absence of 'love triangles', the graph G will be consistent.*

The essential reason for generating the 'love triangles' in a graph, mostly in candidacy graphs, is that certain nodes are over-loaded with multiple labels and thus they are coupled with conflicting nodes. For example, the node 'eye' should be either a 'rabbit eye' or a 'duck eye' and it should be split into two conflicting candidates connected by an negative edge. This way it can eliminate the "love triangle". Figure 6.13 illustrates that we can remove the love triangle by splitting node 1 into nodes 1 and 1' and thus we will have consistent ccp .

6.5.2 Formulation of Hierarchical C^4

One other issue that we need to address is higher-order constraints that involve more than 2 nodes. Such constraints are very common in grammatical, compositional, and hierarchical models where visual entities are decomposed into multiple constituent

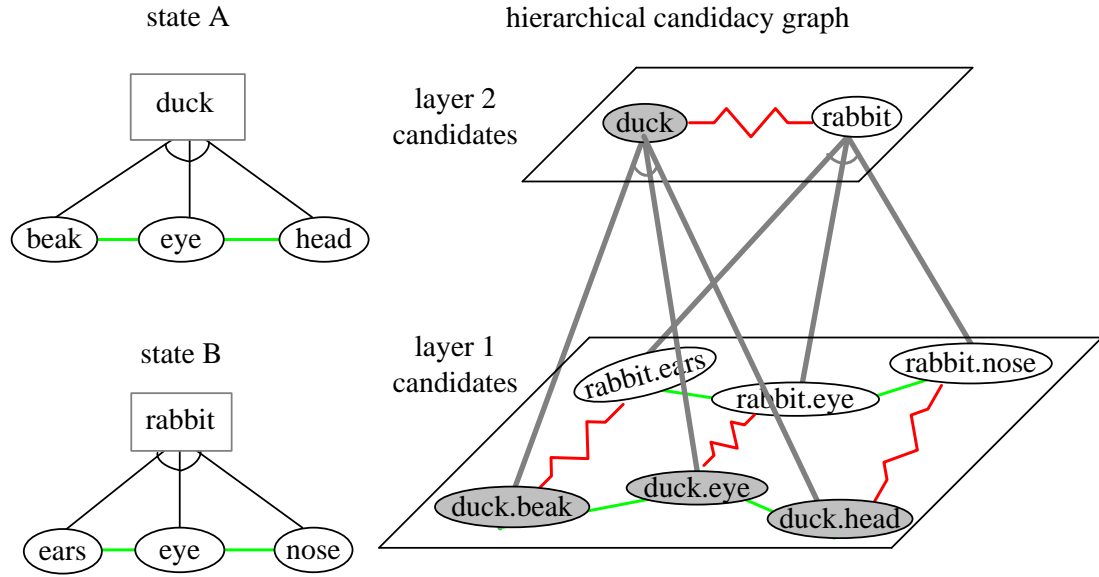


Figure 6.14: An attempt to solve the duck/rabbit illusion using hierarchical C^4 . The trees define which parts comprise each object. Nodes are grouped according to these trees, creating higher-level nodes. The higher-level nodes inherit the negative constraints.

parts. Figure 6.14 shows a hierarchical graph representation for the duck/rabbit illusion. This is a candidacy graph with two layers. The top layer contains two hidden candidate hypotheses: 'duck' and 'rabbit'. The two nodes are decomposed into three parts in layer 1 respectively and thus impose high order constraints between them. Now the hypotheses for parts are specifically for 'duck.eye', 'rabbit.eye' etc. The negative edge connecting the two object nodes is inherited from their overlapping children.

This hierarchical candidacy graph is constructed on-the-fly with nodes being generated by multiple bottom-up detection and binding processes as well as top-down prediction processes. We refer to a recent paper by Wu and Zhu (Wu and Zhu, 2010) for the various bottom-up/top-down processes in object parsing. In this graph, positive and negative edges are added between nodes on the same layers in a way identical to the flat candidacy graph, while the vertical links between parent-child nodes are

deterministic.

By turning on/off the positive and negative edges probabilistically at each layer, C^4 obtains *ccp*'s and *cccp*'s as in the flat candidacy graphs. In this case, a *ccp* contains a set of nodes that are coupled in both horizontal and vertical directions and thus represents a partial parse tree. A *cccp* contains multiple competing parse trees, which will be swapped in a single step. For example, the left panel in Figure 6.14 shows two *ccp*'s for the duck and rabbit respectively which are connected with negative edges in the candidacy graph.

This hierarchical representation can also eliminate the inconsistency caused by overloaded labels. That is, if a certain part is shared by multiple object or object instances, we need to create multiple instances as nodes in the hierarchical candidacy graph.

6.5.3 Experiments on Hierarchical C^4

To demonstrate the advantages of hierarchical C^4 over flat C^4 , we present experiments on interpreting the duck/rabbit illusion and finding configurations of object parts amidst extremely high noise.

Hierarchical Duck/Rabbit Illusion To demonstrate the advantages of hierarchical C^4 over flat C^4 , we present an experiment for interpreting the duck/rabbit illusion. We will use C^4 for finding configurations of object parts amidst extremely high noise using our models in Chapter 7.

Experiment on Hierarchical Duck/Rabbit Illusion. As referenced above, C^4 on the flat candidacy graph in Figure 6.12 creates two love triangles. As such, when the algorithm converges, it selects the entire graph as a proposal structure, which it cannot determine a suitable labeling for. This conundrum is familiar to other graph inference

problems, as we found that graph cuts could not find a labeling for this graph as well. The top panel of Figure 6.15 shows the results of flat C^4 on the duck/rabbit illusion. C^4 continuously swaps between two states, but the two states either have all nodes on or all nodes off, neither of which are valid solutions. The bottom panel of Figure 6.15 shows the results of applying hierarchical C^4 to the duck/rabbit illusion. We defined a tree for the duck/rabbit illusion consisting of either a duck, $\{beak, eye, duck.head\}$, or a rabbit $\{ears, eye, rabbit.head\}$. As a result, the algorithm instantly finds both solutions and then proceeds to swap between them uniformly. These results show that hierarchical C^4 can help guide the algorithm to more robust solutions and negates the effects of love triangles.

6.6 Conclusions on C^4

In this chapter we presented C^4 , an algorithm that can handle complex energy minimization tasks with soft and hard constraints. By breaking a large CSP into smaller sub-CSPs probabilistically, C^4 can quickly find multiple solutions and switch between them quickly. This combination of cluster sampling and constraint-satisfaction techniques allows C^4 to achieve a fast mixing time, out-performing single-site samplers and techniques like belief propagation on existing problems. This novel algorithm can sample from arbitrary posteriors, and is thus applicable to general graphical models, including MRFs and CRFs. In addition, we were able to use a hierarchical prior to guide our search to avoid frustrations in the graph and thus achieve richer and more accurate results than just by using Flat C^4 alone. In the next chapter we will see applications of C^4 to our object and aerial image models.

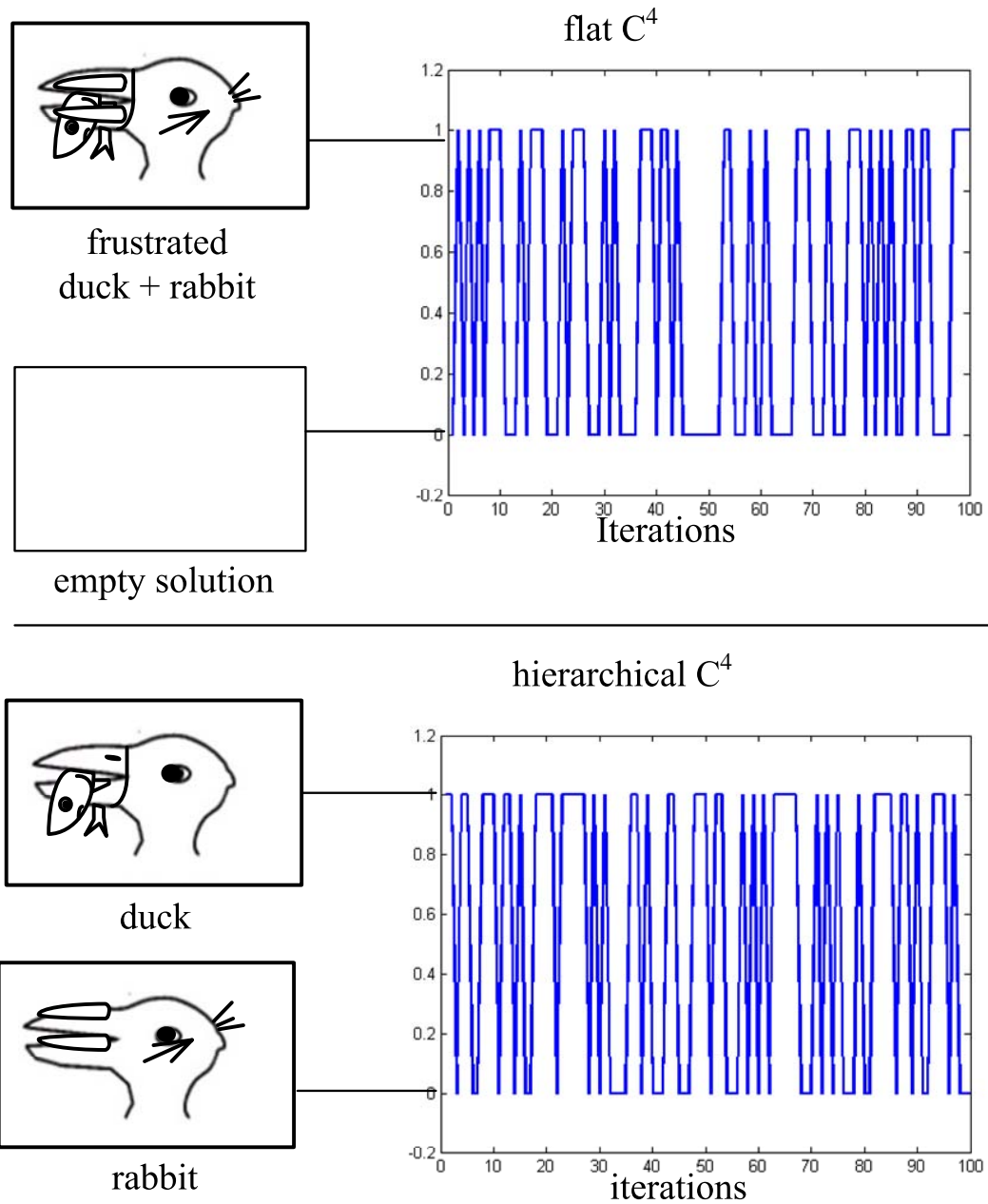


Figure 6.15: (Top panel) Flat C^4 results on the duck/rabbit illusion. C^4 swaps between two impossible states due to love triangles. (Bottom panel) Hierarchical C^4 results on the duck/rabbit solution. C^4 now swaps uniformly between the two correct solutions.

CHAPTER 7

Experiments on Inference

Given that we’ve learned our hierarchical contextual models for objects and aerial images as in Chapter 4, we now describe bottom-up detection methods combined with C^4 for parsing images using our inference methods.

7.1 Object Inference

A problem that often appears in computer science is the problem of finding the optimal subset from a larger set of items that minimizes some energy function. In computer vision, this problem often presents itself as finding the best subset of detections that correspond to true detections. For example, in the constellation model, a parts-based object model, many instances of each object part may be detected in the image (e.g. a car detector may find 5 wheels, 3 bodies, and 4 windshields). However, our algorithm should find the subset (or subsets) of these detections that creates the highest probability configuration (e.g. the 4 wheels, 1 body, and 1 windshield that look most like a car). This is a combinatorially hard problem as the number of solutions grows exponentially in the number of detections, so heuristic approaches are usually proposed to deal with this situation.

Hierarchical C^4 is ideally suited for this problem, as it can use local edge constraints and hierarchical grouping to guide its search through large sets of detections to find the most likely solutions. In this experiment, we learned our model for 4 cate-

gories of objects: 'side car', 'front car', 'teapot', and 'clock'. We then present C^4 with a set of detections as noisy distractors, within which a true object was included. For each true detection we added 50 false detections at random orientations, scales, and positions, as shown in Figure 7.2. If we only considered configurations of 4 parts, finding the optimal configuration would require exhaustively searching 64,684,950 configurations, in the worst case, which quickly becomes intractable when considering larger configurations or more detections. Because our object parts for the object model are graph structures that are difficult to detect, we generate the false distractors and true positives in this case. Later work will examine terminals that can be generated using detection algorithms.

To create our candidacy graph we let each bottom-up detection be a vertex in the graph, connected by edges with probabilities proportional to how compatible those objects are. Each candidate can be on or off, indicating whether it is in the current explanation of the scene or not.

Each edge is assigned to be positive or negative and assigned a probability q_e of being on by examining the energy $e = \sum_{i=1}^{N(R)} \langle \lambda_i^{(\beta)}, H_i(x_s, x_t) \rangle$ between its two nodes (x_s, x_t) . If $e > T$, the edge is labeled as a negative edge and if $e < T$ the edge is labeled as a positive edge, where T is a threshold of the user's choosing. In our experiments we let $T = 0$. Next, we apply a squashing function to the pairwise energy to normalize the energies to the range $[0, 1]$. In our experiments we used a logistic function $F(e) = \frac{1}{1 + \exp\{-(e-T)/s\}}$, where T is our threshold from above and s is a scale parameter that determines how soft or hard the edges are. We then apply $F'(e) = 1 - F(e)$, $e < 0$ to make our probability function symmetric for both positive and negative edges, then rescale to $[0, 1]$ via $F''(x) = 2 * F(x) - 1$. This creates a symmetric function where a value much less than t is a positive edge with probability close to 1, and a value just slightly larger than T is a negative edge with probability

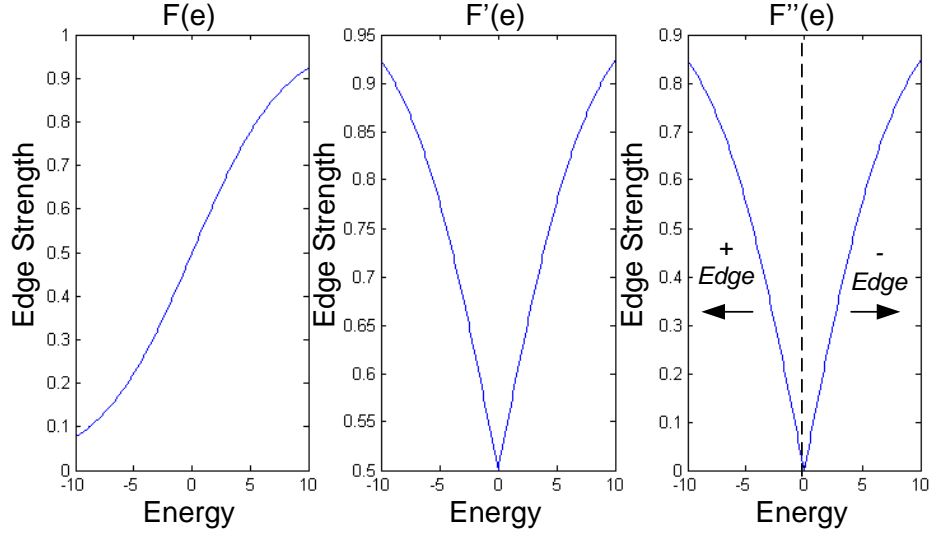


Figure 7.1: The edge strengths for positive and negative edges. The energy between two nodes $F(e)$ is computed based on the learned model and then reparameterized using $F'(e)$ and $F''(e)$ to create a function that maps an edge type and probability for each pairwise energy. Parameters are $T = 0, s = 4$

close to 0. The value s determines how quickly the edge probability saturates to 1 as e moves away from T . Figure 7.1 visualizes this function with $T = 0$ and $s = 4$. Using the squashing function we create data-driven edge probabilities and determine positive and negative edge types for C^4 .

To test both C^4 and our learned models we compare to Iterated Conditional Modes, Swendsen-Wang, C^4 , and Hierarchical C^4 . In each scenario the detections can either be “on” or “off”. Each inference algorithm then attempts to use the unary probabilities and the pairwise positive/negative edges to determine the best subset of detections, if any, for each image. We could not compare to graph cuts because it could not return an answer on these types of graphs.

The results of finding the optimal subset using ICM, Swendsen-Wang, flat C^4 , and hierarchical C^4 for the teapot category are shown in Figure 7.2. We can see that

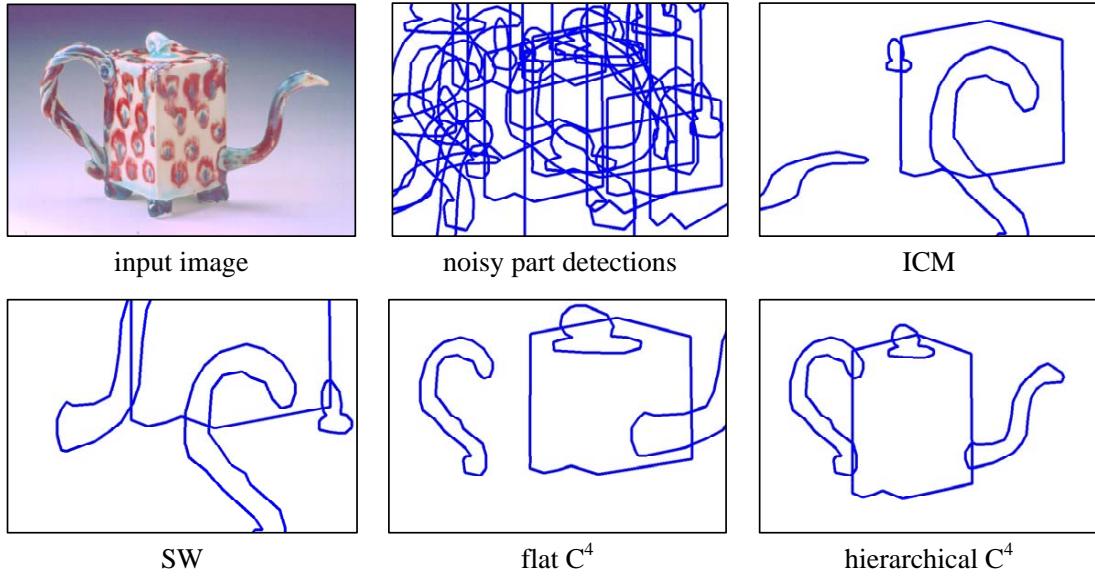


Figure 7.2: Hierarchical C^4 for detecting signal from noise. A huge set of distractors are added over a true parse of an object. Using our learned model, C^4 can find the best subset while other algorithms cannot.

ICM and Swendsen-Wang find completely unreasonable solutions. This is due to the fact that they quickly get stuck in sub-optimal solutions. Flat C^4 does not find the correct solution, although it does find a set of parts that are fairly low energy, as the resulting image does look similar to a teapot. Hierarchical C^4 , on the other hand, quickly converges to the true solution amidst the myriad other possible part combinations available.

Figure 7.3 shows the results of other signal-from-noise images that were generated as above. We show the results divided roughly into three categories: good, medium, and bad results. A good result selects all of the correct parts for each image, a medium result selects 1-2 parts wrong, while the rest are considered bad. The righthand graphs show the proportion of the testing images that were good, medium, or bad according to algorithm. We see that hierarchical C^4 gets mostly good results, while ICM gets

entirely bad results. Flat C^4 achieves a fair proportion of good results, but not nearly as many as Hierarchical C^4 does.

Figure 7.4 shows the energy of the system over time for the four algorithms we tested. We can see that, not only does hierarchical C^4 achieve a minimum energy almost instantaneously, but that both hierarchical C^4 and flat C^4 are able to achieve lower energy minimums than the other methods. This improvement applies to graph cuts as well, which, as mentioned, are not shown here because no implementation we found was able to converge in the presence of love triangles. This result shows Hierarchical C^4 ’s ability to quickly find deeper energy minima than competing approaches.

In our experiments we found that, in certain cases, the energy function being minimized could be tweaked such that flat C^4 would actually find the correct solution more frequently. However, aside from the inconvenience of adjusting energy functions, we found that (i) flat C^4 could not always find the correct solution, and (ii) flat C^4 was much slower than hierarchical C^4 , requiring 15 seconds to parse an image that Hierarchical C^4 could parse in under 1 second.

These results show the power of Hierarchical C^4 for quickly finding minimal energy subsets and swapping between equally or nearly-equally likely solutions once found, where as similar methods (Swendsen-Wang, ICM, Graph Cuts) fail to even find a viable solution. It also shows that our learned model enforces correct part arrangements, as Hierarchical C^4 finds the true object boundaries according to the model energy quite easily.

7.2 Aerial Image Inference

In this section we show the results of using C^4 and our learned aerial image model for aerial image parsing. Unlike the object detection results that required hand-generated



Figure 7.3: Examples of good/medium/bad results for Hierarchical C^4 , Flat C^4 , Swendsen-Wang cuts, and iterative conditional modes (ICM). The graphs to the right show the proportion of the testing images that belonged to each ranking according to algorithm.

detections, we have a number of methods for detecting each type of object in an aerial image. We then use C^4 to find the best subset of detections that describe the aerial image in accordance with our bottom-up probabilities and our top-down model. We first describe our bottom-up detection methods and then present results of applying C^4 for inferring the best parse of the scene.

7.2.1 Bottom-Up Detections

We first use bottom-up detectors to find initial detections for each type of object in each new aerial image:

Cars: We trained a discriminative AdaBoost classifier (Freund and Schapire, 1997) to detect cars. We collected 3000 positive examples of cars, selected by hand as patches containing a single car from aerial images, as well as 3000 negative images for training, comprised of patches of training images in which no car is present. Figure 7.5(a) shows car detections using the learned classifier. Unfortunately, we do find that this method results in many false positives, which we will address later. AdaBoost

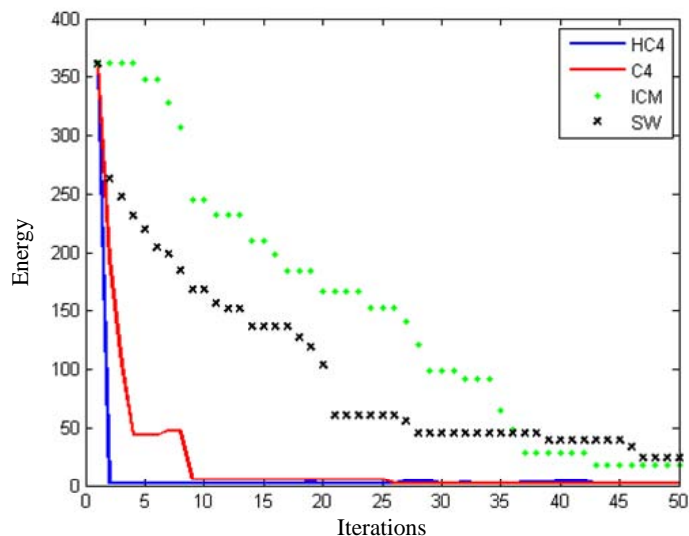


Figure 7.4: Plots of energy over time for Hierarchical C^4 , Flat C^4 , Swendsen-Wang cuts, and iterative conditional modes (ICM). Not only does Hierarchical C^4 converge fastest of all the algorithms, but it achieves a lower energy than the other methods.

is a commonly cited and described algorithm, so we refer the reader to (Freund and Schapire, 1997) for further details.

Parking lots and Trees: Parking lots and trees are characterized by their textures. Color information is highly variable from one parking lot or grove of trees to the next, so color histograms are too simple to capture an appearance model for these classes. We resolve this problem by using TextonBoost (Shotton et al., 2006), an algorithm for combining texture and shape cues in a boosting framework to create a discriminative classifier. TextonBoost extracts textons (collections of filter responses) for each category and clusters them into a texton dictionary. These textons are then boosted using to arrive at a combined discriminative classifier. We provided TextonBoost with about 100 images in which the images are labeled (0/1) according to whether or not a pixel belongs to background or the category we're learning (parking lots and trees are

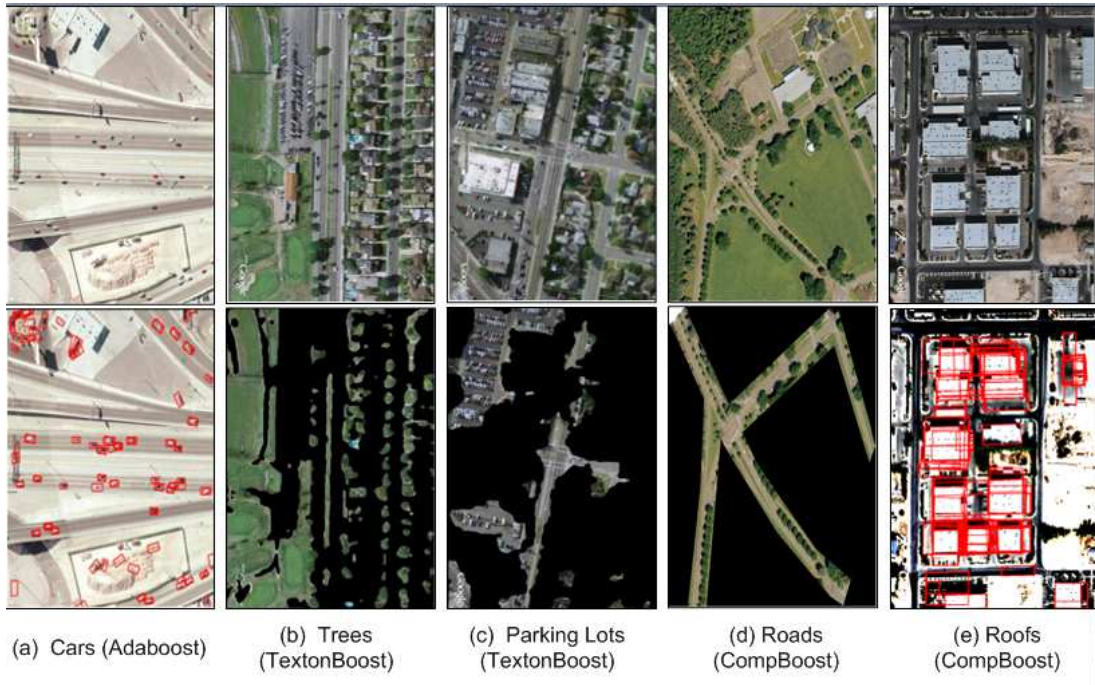


Figure 7.5: Single object detections using our bottom-up detectors.

learned separately). The pure bottom-up results are shown in Figure 7.5(b) and Figure 7.5(c).

Roofs and Roads: Roofs and roads present quite a different problem from the categories we’ve represented up until now. They are neither defined by a constant shape nor a constant texture. The most informative cues are the edges that define their boundaries. We use a recently developed algorithm called Compositional Boosting (Wu et al., 2007) that hierarchically combines low-level cues into higher-level structures. A more detailed explanation of Compositional Boosting is given in (Wu et al., 2007), but we will describe it at a high level here. Results for road and roof parsing are shown in Figure 7.5(d) and (e).

7.2.1.1 High-level Description of Compositional Boosting

Compositional Boosting learns a model by first defining a dictionary of low-level features (such as edges) along with some spatial rules of interest (e.g. parallelism, relative length, collinearity). These low-level features are first labeled in a number of training images and labeled as belonging to the structure of interest or not. For example, in this experiment, we labeled edges in the training data as belonging to a roof, a road, or neither. Compositional Boosting begins building a hierarchy from these labeled edges by testing the mutual information of edges under certain spatial constraints. For example, in the roof class we will see lines at right angles more frequently than in random noise. Any composition rules with mutual information greater than some threshold are added to the hierarchy (e.g. two lines nearly 90 degrees from one another should form a higher-level component). This process then repeats at the next highest level until some percentage of the labeled lines are modeled by the final composition. Figure 3.6 shows the Compositional Boosting hierarchies below the roof and road nodes. A roof can decompose into a number of different shapes, each of which is formed from lower-level components.

To detect structures in images, we first define detectors for Compositional Boosting. We begin with an edge detector for edges, since they are the lowest level nodes in our hierarchy. However, we may also define higher level detectors to find higher-level nodes (e.g. corner detectors). Let us define a possible set of detectors $T = \{t_i : i = 1, 2, \dots, k\}$ at each node designed to detect that part directly from the image. We also add auxiliary data structures to each node, called “Open” and “Closed” lists. The open lists will store any current potential detections for that node. The closed list will store any accepted detections of the node the list resides at. Each proposal in an open list is weighted by a posterior probability ratio.

Compositional Boosting first creates proposals for the open lists for an image I in

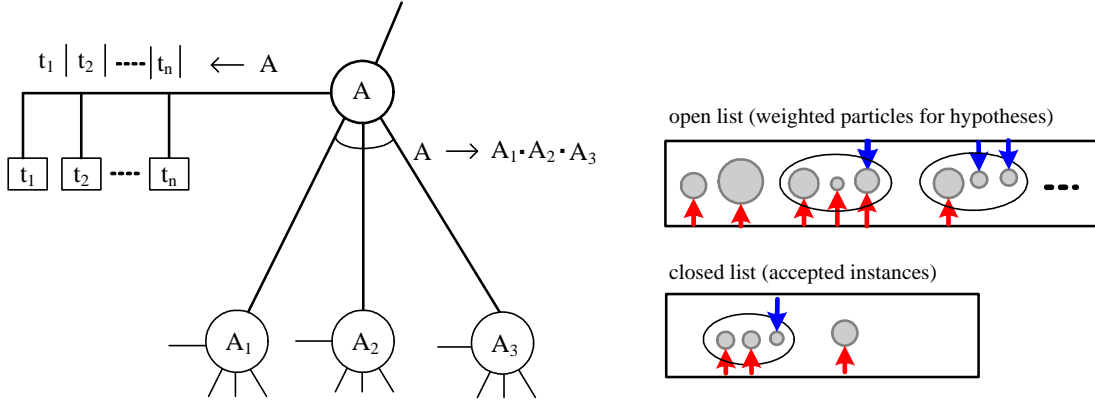


Figure 7.6: A conceptualization of inference with Compositional Boosting. The left hand side shows an example of a node in a Compositional Boosting tree with parent node A and children nodes (A_1, A_2, A_3) . (t_1, \dots, t_n) indicate proposals for A detected directly from the image, while $A = A_1 \cdot A_2 \cdot A_3$ indicates proposals for A detected as a product of child proposals. In the inference process, we store proposals at node A in open and closed lists, where particles in the open list are pending proposals and particles in the closed list have been accepted. The red and blue arrows in the figure indicate that there is evidence for each particle coming from both bottom-up and top-down channels.

one of two ways:

(1) Proposals for A are formed from local detectors T . The weight of each detection is the log-ratio of the local marginal posterior probability on an image patch λ_i using some features of the image $F()$,

$$\hat{\omega}_A^i \approx \log \frac{p(A^i | F(I_{\lambda_i}))}{p(\bar{A}^i | F(I_{\lambda_i}))}, \quad (7.1)$$

where \bar{A} is an alternative hypothesis.

(2) Proposals for A are formed by combining proposals for A 's children from their Open and Closed lists. Proposals from each list are compared based on their compatibility, and highly compatible proposals are combined to propose the higher level node

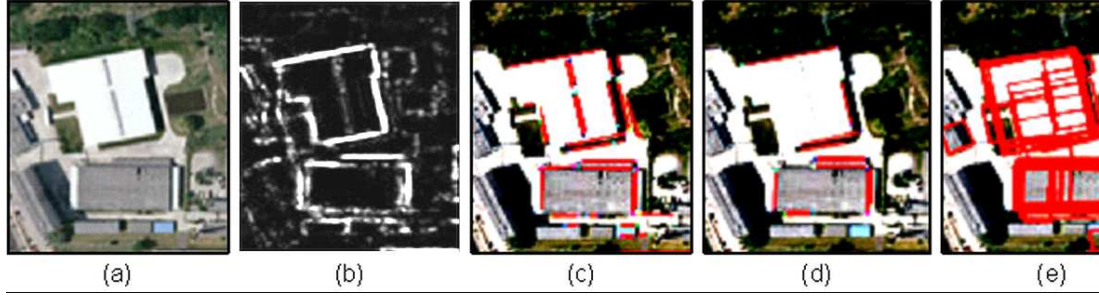


Figure 7.7: An example of detecting roofs with Compositional Boosting. We run a probabilistic edge detector to get image (b), after which the algorithm detects object parts, such as parallel lines and corners in (c) and (d). These act as evidence for the final roof proposals in (e).

A. The weight on these hypotheses is the local conditional posterior probability ratio. Suppose a proposal A^i is formed from three of its child proposals A_1^i , A_2^i , and A_3^i , then the weight will be

$$\hat{\omega}_A^i \approx \log \frac{p(A_1^i, A_2^i, A_3^i | A^i) p(A^i)}{p(A_1^i, A_2^i, A_3^i | \bar{A}^i) p(\bar{A}^i)} \quad (7.2)$$

where \bar{A} represents a competing hypothesis. In other words, we are measuring the probability that these proposals appeared as a result of A existing as opposed to some other node. The top-down process then greedily adds proposals from the Open lists to the Closed lists and updates the Open list weights until no weights are above a certain threshold.

Figure 7.6 shows a toy example of this process. Here we see that A can be formed from either its detections T , or by combining proposals from its children. This is where Compositional Boosting is particularly powerful, because weak detections of compatible children may be enough for us to propose the parent node.

Figure 7.7 shows an example of roof detection using Compositional Boosting. We begin with a probabilistic edge map formed from our source image. From this map

we first extract edge segments using an edge detector. We next combine edges that are compatible according to the rules that we’ve learned from labeled roofs. Figure 7.7(c) shows parallel lines detected in the image, while Figure 7.7(d) shows corners detected in the image. Figure 7.7(e) shows the final roof detections inferred from the composition of these low-level features. We can see that the rectangular structures of roofs are detected, but there are also many false positives present.

7.2.2 Top-Down Prediction of Missing Objects

Because we have a generative model for scenes, we can use the results from the first round of initial bottom-up detections to suggest where other objects may have been missed. For example, if we detected four cars in a row with a gap in between them, it might be reasonable to predict that another car should be present there. Figure 7.8 shows an example of predicted roofs, cars, and roads based on our results from stage two and our prior model. The hallucinated objects are shown in green dashed rectangles, while the accepted detections from C^4 are shown in black solid rectangles. These top-down predictions will then be pruned or accepted using a final round of C^4 .

Our top-down prediction method consists of sampling new parse graphs based on our original parsing results. We perform C^4 to determine which objects are most likely good detections and form a parse graph for the scene. We then sample new node frequencies and part relationships for the parse graph, accepting them probabilistically using a Metropolis-Hastings step. For example, our parse graph may describe a row of three cars with a large gap in between them. Our model may find a lower energy arrangement describing the scene by five cars where the extra two cars fill in the gap. Once this new parse graph is proposed, we verify the top-down detections by giving them a uniform detection probability and then running another round of C^4 .

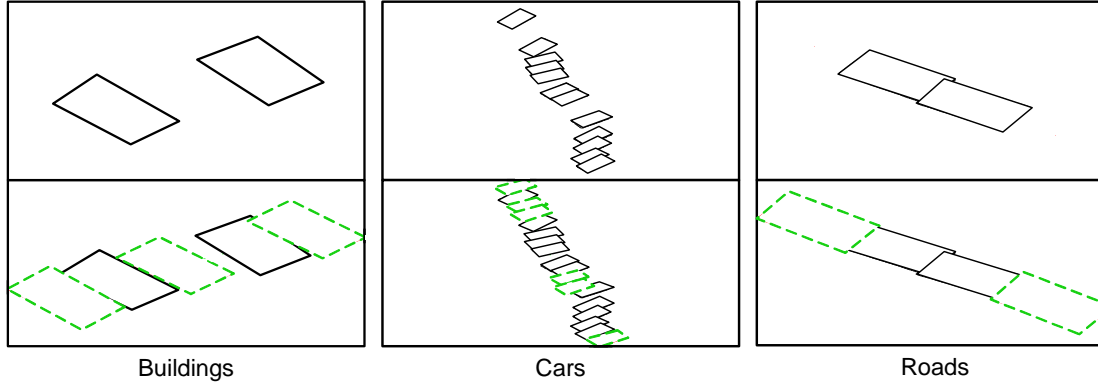


Figure 7.8: Top-down hallucinations of missing objects. Black rectangles indicate the detections from our first run of C^4 , while the green dashed rectangles indicate hypotheses for missing objects proposed by the top-down part of our model.

7.2.3 Experimental Results

We ran our algorithm on 5 large (4000x4000) images collected from Google Earth. We learned a top-down model as in Chapter 4 and implemented detectors for each of the objects as described in Section 7.2.1. Figure 7.9 shows the process of our algorithm on one aerial image. The first panel shows the original image, while the second panel shows an overlay of the initial bottom-up detections, which contains a huge number of false positives (3 false positive roads, 71 false positive buildings, 623 false positive cars, 10 false positive trees). The third panel shows the results of using C^4 clustering to find a high probability set of bottom-up detections to explain the scene. The fourth panel shows the final explanation of the image after some new proposals have been suggested and verified. The first step of the C^4 algorithm shows the most dramatic improvement, with vast numbers of inconsistent detections (cars on roofs, trees on roads, overlapping roofs) being removed, leaving just single object boundaries for the important objects (we now have 0 false positive roads, 5 false positive buildings, 57 false positive cars and 0 false positive trees). The second step gives a slight improve-

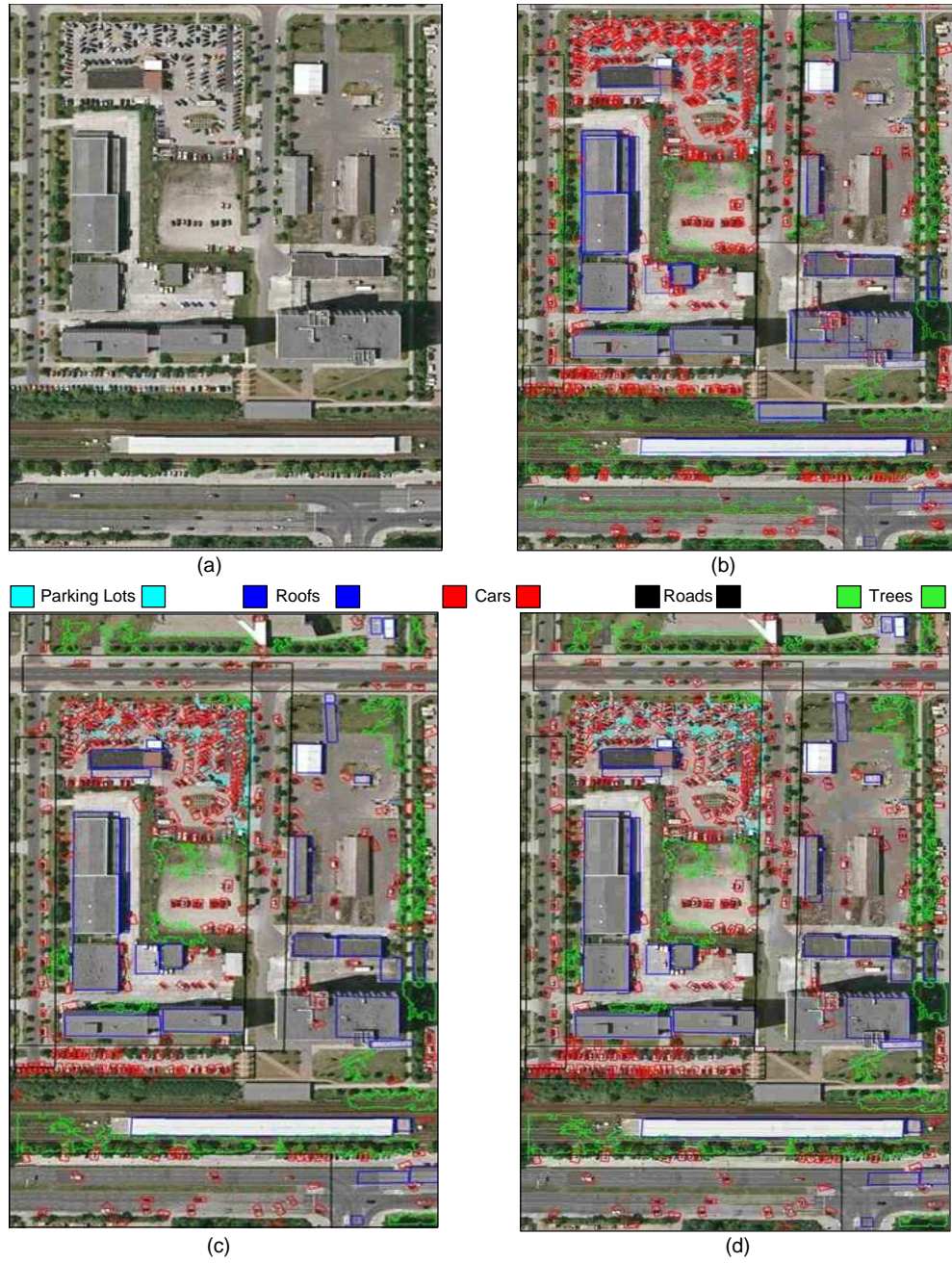


Figure 7.9: The bottom-up to top-down pipeline. (a) The original image. (b) The bottom-up detections. There are a huge number of overlapping and inconsistent detections. (c) The top-down pruning results using C^4 . Many false positives are removed. (d) The results given newly proposed nodes from the hierarchical prior.

ment, though primarily just in finding missing cars, which are difficult to see at this resolution. Note that there are still some missed detections, either because our initial detectors did not detect the object or because the context may have inadvertently ruled out a valid explanation (e.g. accidentally favoring the shadow of a roof instead of the roof itself, thus suppressing the true roof).

The inference stage, given bottom-up proposals, takes about 10 seconds to run on a dual core 1.6GHz machine. The bottleneck in our pipeline is the detection phase, however. For example, our AdaBoost results take a mere couple of seconds to compute. The edge detector we used, on the other hand, can take upwards of a minute to process each image. Therefore, the speed of our approach is highly dependent upon the speed required to compute the initial bottom-up detections.

7.2.3.1 Compositional Boosting Results

Because of the newness of Compositional Boosting, we first examined how much improvement we achieved in detecting low-level roof parts using Compositional Boosting. Figure 7.10 shows ROC curves for detecting U junctions, L junctions, parallel lines, and opposing L junctions. Using specific bottom-up detectors alone (the blue curves) causes us to miss a lot of the junctions present. By using Compositional Boosting, we are again able to leverage context and hierarchy to identify missing junctions to help us propose more roofs, as shown by the red curves.

7.2.3.2 Top-Down Pruning

Figure 7.11 shows a zoomed in view of our test images before and after pruning. Figure 7.11 shows that, at first, we have many conflicting proposals for the object boundaries, notably that a parking lot could be on top of the roof. After we enforce the contextual constraints we learned, however, we return to a sensible explanation of the scene, one

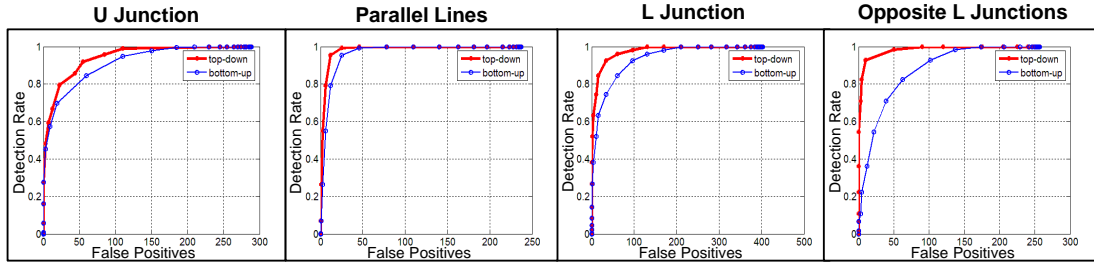


Figure 7.10: ROC curves for detecting U junctions, L junctions, parallel lines, and opposing L junctions using Compositional Boosting. We see that CompBoost helps us identify weakly detected junctions, which helps us propose better high-level detections.



Figure 7.11: Close up views of our improvement during pruning. Notice that overlapping proposals and inconsistent explanations (cars in trees) have been removed.

in which there are no longer cars on top of roofs or overlapping proposals. Figure 7.12 shows a zoomed in view after we propose new cars. Initially we missed some cars in the rows of the parking lots. Because our model recognizes that cars appear in rows, however, it proposes cars of roughly the same shape and sizes of the neighboring cars around them, using a line grammar. Cars matching above a certain likelihood are accepted and the conflicting nodes are removed.

Table 7.1 shows the detection rate and false positives per image of each category using just that category’s detector (shown in parentheses) vs. using the full hierarchical



Figure 7.12: Close up views of our improvement during top-down prediction. Additional cars are added to the rows due to the presence of other collinear cars.

contextual model. We can see that the hierarchical contextual model greatly reduces the false positive rate from single-object bottom-up detectors because it can leverage context to remove false positives. Our detection rate is about the same, however, as the pruning phase in the second stage serves mostly to rule out inconsistent detections. In the third stage we were able to identify a few extra cars (as shown in Figure 7.12), but the amount of extra detections was not enough to account for the inadvertent pruning of true positives from stage 2. Overall, our context allows us to achieve comparable detection rates to single-object detectors, but with far fewer false positives.

Figure 7.13 shows two different precision-recall curves for the bottom-up and top-down stages of our process. We show precision-recall as opposed to ROC curves because it is difficult to decide how to compute the number of true negatives for multi-category classification tasks, a decision that can drastically alter the appearance of the algorithm’s performance. In Figure 7.13(a), we measure our accuracy at the pixel level. In this experiment, we labeled each pixel in the image as belonging to an object category or not and then converted our inferred boundaries to a similar labeling. In Figure 7.13(b) we measure our performance using object-level accuracy. In this case we considered an object to be detected if it had a boundary around it within some threshold of its true scale and position. In both cases we can see that the top-down

Table 7.1: False positives per image and detection rates for bottom-up detectors versus our method.

Detection method	False positives per image	Detection rate
Cars (AdaBoost)	242.33	88.1%
Cars (Ours)	71.83	84.2%
Parking Lots (TextonBoost)	1.17	84.3%
Parking Lots (Ours)	0.16	84.3%
Trees (TextonBoost)	14.5	88.8%
Trees (Ours)	9.33	88.8%
Roofs (CompBoost)	73.5	70.3%
Roofs (Ours)	1.67	70.3%
Roads (CompBoost)	5.67	95%
Roads (Ours)	0.05	88.3%
Combined (All Detectors)	337.17	93.1%
Combined (Ours)	83.04	87.5%

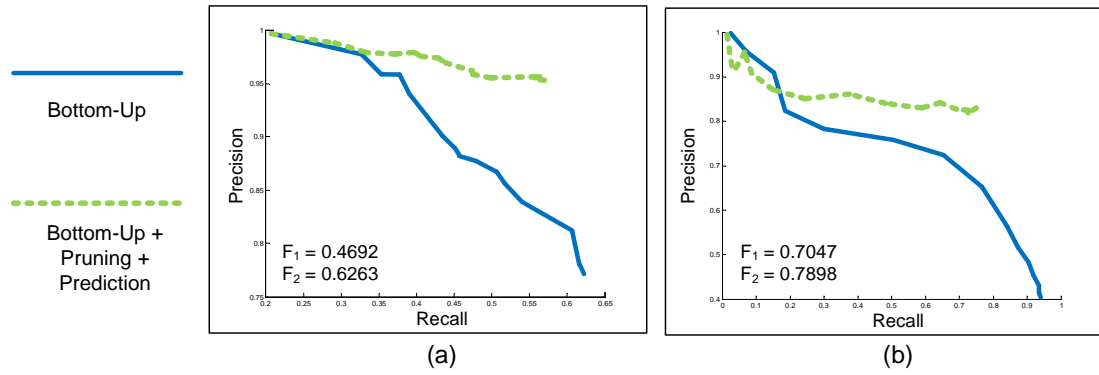


Figure 7.13: Precision-Recall curves for the bottom-up and top-down inference algorithm. In both cases we see a huge improvement by using C^4 to prune out false positives and using our model to predict missing objects. F_1 and F_2 are the best F-measures for the bottom-up and for the full algorithm, respectively. (a) Precision-Recall curve using pixel-level accuracy, i.e. each pixel in the image is assigned a category label. (b) Precision-Recall curve using object-level accuracy, i.e. each object is considered detected if we infer an object of appropriate dimensions over it.

improvements over the initial detections are substantial. While the initial detections give average performance, it is the introduction of the top-down pruning and prediction that flattens our curve, enabling us to keep a very high level of precision as the recall increases. Notice, however, that in Figure 7.13(b) the second stage actually degrades performance slightly for low values of recall, likely because it has pruned too many true positives, reducing our precision slightly. This could likely be improved by adjusting the likelihoods of our initial candidates so that we don't overprune them. Overall though the top-down performance far eclipses the initial bottom-up detection results on their own.

We also looked at the many benefits of using C^4 to find solutions. Figure 7.14 shows an example of a patch where C^4 is extremely useful during inference. In the first panel, the algorithm has mistakenly interpreted the vents on top of the building as

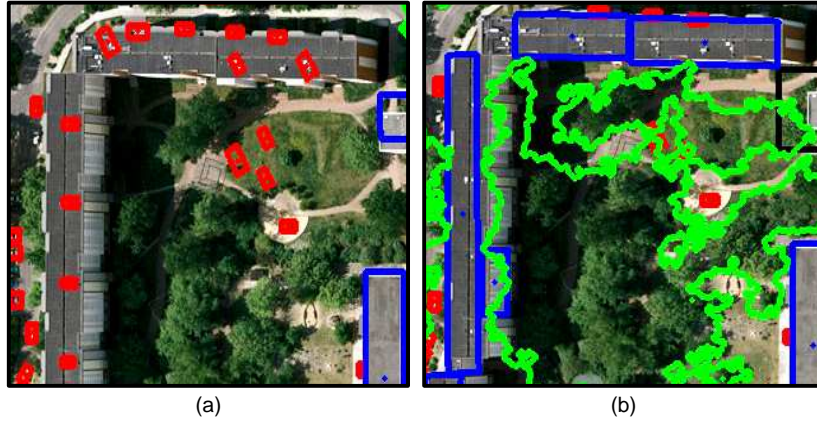


Figure 7.14: An example of swapping alternative solutions using C^4 . (a) Vents are incorrectly labeled as cars on top of the roof. (b) The cars are correctly swapped out for the roof simultaneously to arrive at the correct solution.

cars. This has thus ruled out the true explanation that there is a building there. Thanks to C^4 's ability to swap out all of the related cars while simultaneously adding the roof, we are able to arrive at the correct solution in panel (b). This solution is maintained because it has a higher probability than the previous explanation.

7.2.3.3 Model Complexity Results

In Section 4.3 we mentioned that we choose to add relationships iteratively instead of fitting the full model, as this allows us to keep our model simple for sampling and performing inference. The question remains, however, about whether we need to learn as many relationships as we do. Figure 7.15 shows the inference results for an aerial image using a model with a high ε_1 (10) and a model with a standard ε_1 (4). We can see that the partially learned model is lacking contextual relationships for cars and buildings, as many cars appear on roofs, and cars appear on top of one another. The fully learned model does not make these mistakes. While ε_1 is definitely variable, we strive to select a value that produces good results while still minimizing the size of our

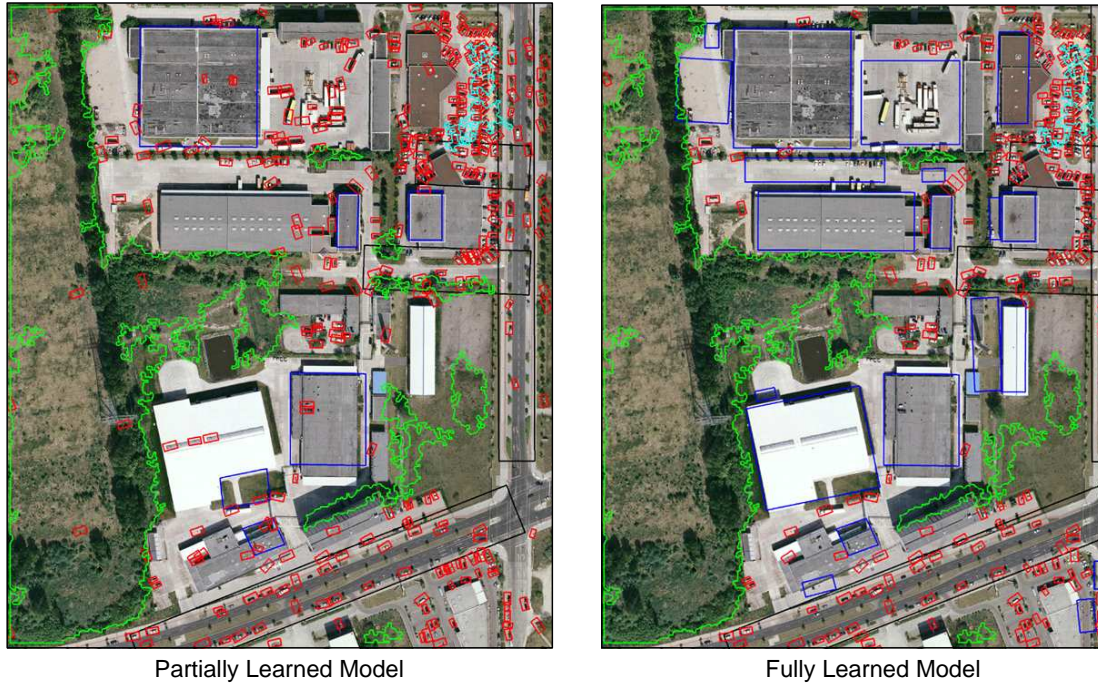


Figure 7.15: A comparison of inference results for the model learned with a very high ε_1 versus a rather low ε_1 (i.e. fewer relationships are added to the first model). The partially learned model is missing a lot of overlap constraints (e.g. cars on trees, cars on buildings), and so makes very poor decisions when parsing the scene. Many of the buildings have cars on top of them and cars readily overlap each other.

relation set R .

Figures 7.16 and 7.17 show the final results of our algorithm on a number of other urban aerial images. We used our algorithm to find the best parse graph representations for each object and here just display the flattened configurations of the highest probability parse graph for each scene. We can see that the majority of objects are detected accurately, though there are still a few false positives.

We would like to compare our methods to other works in the field, but, as mentioned in Section 6.1, we are hard-pressed to find competing algorithms that iden-

tify multiple categories of objects. Similarly it was quite difficult to find benchmarks on consistent datasets in the aerial imaging community, so we would like to offer these results as a benchmark on the aerial images we used from the Lotus Hill Database. These images will be available from the Lotus Hill Institute’s website (<http://www.imageparsing.com>) and can be used freely by anyone else interested in testing on them.

7.3 Conclusions on Inference Experiments

In this chapter we showed the power of our model and inference algorithm for parsing new images. We saw that Hierarchical C^4 was able to find the correct subset of object parts very quickly even in very noisy situations. We also showed that bottom-up and top-down methods could be combined to more accurately parse aerial images than using one source of information alone. We saw that C^4 ruled out many of the inconsistent detections and was able to swap between different interpretations. We were also able to show a very slight improvement in our detection rate by using the model to predict objects that were originally missed by the bottom-up detection phase. Overall we are very pleased with these results and feel that these experiments show both our model’s and our inference algorithm’s power for representing and detecting highly variant image classes.



Figure 7.16: Flat configurations of parsed images.

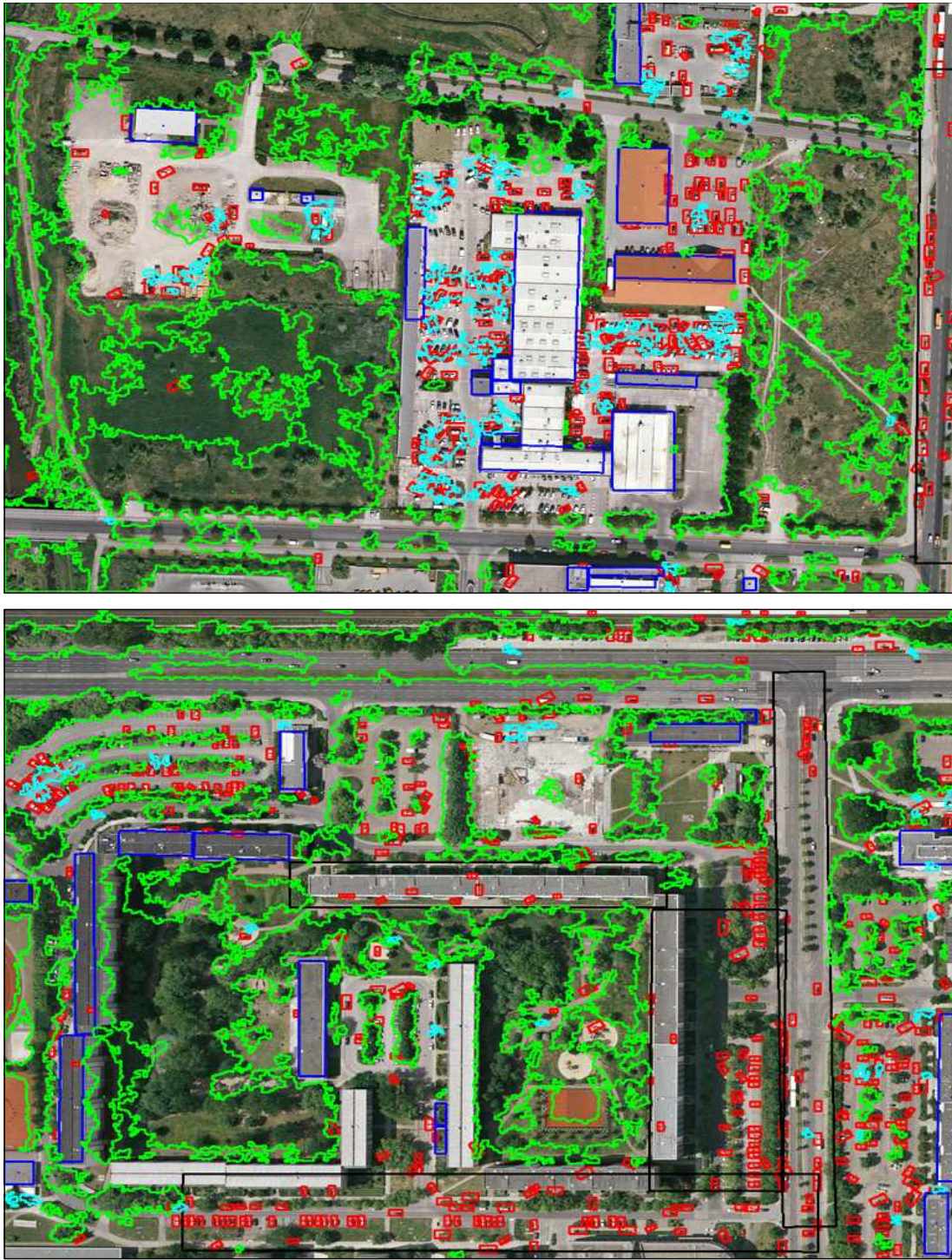


Figure 7.17: Flat configurations of parsed images.

CHAPTER 8

Discussion and Future Work

The main idea introduced in this dissertation is a new generative contextual hierarchy for pattern recognition and modeling. We argue that this combination of hierarchy and context is a critical step forward in modeling highly variant patterns because it combines descriptive models (MRFs for context) with generative models (SCFGs for hierarchy), allowing the model to recognize and recreate a huge range of instances in a very compact framework. To the best of our knowledge, ours is the first model to add horizontal constraints to an SCFG where the constraints can be arbitrarily complex.

We also feel that the minimax entropy learning framework, first used for texture modeling in (Zhu et al., 1997) but applied for the first time here to hierarchical models, is a crucial tool for allowing the system to automatically select relevant relationships for modeling the patterns in question and learning their parameters with a minimum of human intervention. The algorithm that we have laid out can determine what subset of relationships is most important for modeling a given pattern and can then estimate their parameters through iterative sampling of the current model state and comparing it to the real world observed statistics. We showed that iterative pursuit of the relationships provably monotonically decreases the KL divergence between our current model and the target distribution and that estimation of the relationship parameters using our method produces realistic patterns sampled from our model.

Lastly, we feel the C^4 algorithm is a powerful tool for inference in any graph-based model for which interactions can be defined between parts. Current graph cut

and sampling methods are either too slow or unable to cope with multiple solutions and thus cannot solve the type of optimal subset selection problems we are handling. C^4 was shown to very quickly reach a deeper energy minimum and maintain multiple solutions better than any existing algorithms. Moreover, the hierarchical formulation of C^4 can incorporate more complicated priors and can compute more quickly than the flat C^4 representation.

There are many ways in which we will be expanding and augmenting the current model in future work. To begin with, it should be noted that the primitives in the model, which in our experiments were object parts or aerial objects, could be of any form and could have an arbitrary set of attributes. We are therefore interested in extending our work to more general natural images, as well as other domains, such as audio, text, and video. Additionally, we are exploring ways to automatically learn the grammar structure given the observed statistics. It is too laborious to have an operator hand-define a grammar for each pattern class so we therefore need a way to help automate this process. We are exploring existing grammar-building algorithms as well as experimenting with our own methods for constructing these hierarchies. Lastly, we are interested in extending this work to a fully unsupervised framework. Even with the hierarchies automatically generated and the experiments showing that very few training samples are needed to capture the critical parts of an object, it is prohibitively expensive to have humans label training images for each category, particularly if the system is ever to be extended to a general object recognition system. We are therefore interested in extending the grammar-structure down to the primitive level (edges, regions, blobs) and automatically learning the hierarchy from the most basic elements upwards. We will research and develop these ideas in our future work.

8.1 Summary of Major Contributions

The work in this dissertation makes contributions to the fields of machine learning, artificial intelligence, pattern recognition, and statistical modeling in the following ways:

1. The hierarchical and contextual model presented for representing highly variant patterns is a novel representation that captures long-range variability and local context. This model can also be applied to other highly variant data, such as accented speech, genetic sequences, or social network analysis.
2. The application of the minimax entropy learning algorithm to hierarchical models was an advancement over the previous work on solely flat models. This algorithm can be used to quickly and automatically learn context for models where human intervention would be prohibitive.
3. The work on small sample set learning shows that our model can quickly generalize to never-before-seen instances from the same visual category with even as few as six training instances. This makes our model ideal for recognition and discrimination of objects for which it is difficult to obtain large training sets.
4. The ability to incorporate top-down information from the model during inference was shown to improve detection results and brings our research one step closer to a system that can more accurately reason about what it is seeing. The combination of bottom-up and top-down cues are crucial for advanced understanding of new input images.
5. The C^4 algorithm is a novel inference algorithm that can discover and quickly swap between multiple solutions in constrained graphical models. This algorithm can be applied to a huge number of energy minimization problems in

computer vision and machine learning.

Appendix A

Equation 4.20 conjectures that the new relationship r_+ we should add at every iteration should be the one that maximizes the Kullback-Liebler (KL) divergence between our current model $p(pg; \Theta, R)$, (p) and the new model $p_+(pg; \Theta_+, R_+)$, $R_+ = \{R \cup r_+\}$, (p_+) . We first observe that the KL divergence can be viewed as the difference of two expectations:

$$KL(f||p) = \int f \log \frac{f}{p} = \int f \log f - \int f \log p \quad (8.1)$$

$$= E_f[\log f] - E_f[\log p] \quad (8.2)$$

We also learn the λ parameters at each iteration of the learning algorithm such that the expectation of our model matches the expectation of the true distribution for the constraints currently in the model

$$E_p[\log f] = E_f[\log f] \quad (8.3)$$

We can then show

$$KL(f||p) = E_f[\log f] - E_f[\log p] \quad (8.4)$$

$$= E_f[\log f] - E_{p_+}[\log p] \quad (8.5)$$

$$= E_f[\log f] - E_f[\log p_+] + E_{p_+}[\log p_+] - E_{p_+}[\log p] \quad (8.6)$$

$$= KL(f||p_+) + KL(p_+||p) \quad (8.7)$$

Thus, $KL(f||p) - KL(f||p_+) = KL(p_+||p)$, so the relationship r_+ that gives rise to the model p_+ with maximal difference from our current model p will also be the model that maximizes the decrease in divergence between the true distribution f and the new model p_+ .

Appendix B

Estimating $KL(p_+||p)$ in Equation 4.20 is non-trivial, as we have a normalization constant in front of each model that is different for p_+ and p , and each model has a different number of parameters. Equation 4.21 indicates that we can estimate this distance using the Mahalanobis distance between $(H_{r_+}^{(\beta)}(PG^{obs}), H_{r_+}^{(\beta)}(PG^{syn}))$.

r_+ is the new relationship we're adding, h_+ is the expectation of the new histogram H^{r_+} with respect to the new model p_+

$$h_+ = E_{p_+}[H^{r_+}] = E_f[H^{r_+}] \quad (8.8)$$

and h_0 is the expectation of the new histogram H^{r_+} with respect to our current model p

$$h_0 = E_p[H^{r_+}] . \quad (8.9)$$

Because we have not yet added r_+ to the model and reestimated the parameters Θ_+ , $E_p[H^{r_+}] \neq E_{p_+}[H^{r_+}]$.

Our distance measure is

$$D(h_+) = KL(f||p) - KL(f||p_+) = KL(p_+||p) . \quad (8.10)$$

A Taylor expansion about $D(h_+)$ yields

$$D(h_+) = D(h_0) + D'(h_0)(h_+ - h_0) + \frac{1}{2}D''(h_0)(h_+ - h_0)^2 + \epsilon \quad (8.11)$$

and incorporating the higher order terms ϵ into the second-order term gives

$$D(h_+) = D(h_0) + D'(h_0)(h_+ - h_0) + \frac{1}{2}D''(h^*)(h_+ - h_0)^2 , \quad (8.12)$$

where $h_0 \leq h^* \leq h_+$. Because h_0 and h_+ are vectors, we can rewrite the last factor in 8.12 in matrix form:

$$\frac{1}{2}(h_+ - h_0)^T D''(h^*)(h_+ - h_0) \quad (8.13)$$

where $D''(h^*)$ is the inverse of the covariance matrix $cov(h_+, h_0)$. Equation 8.13 can be approximated by the Mahalanobis distance $d_{mahn}(h_+, h_0)$. In order to use that approximation, we must show that

1. $D(h_0) = 0$

2. $D'(h_0) = 0$

$D(h_0) = 0$ must be true because we are not adding any new information to the model by adding h_0 , which is already in our model.

$D'(h_0) = 0$ can be shown by looking at the term $D(h_+)$ in terms of our new p_+ , which consists of all the relationships in our old model p and one new relationship, r_+ :

$$D(h_+) = E_p[\log p] - E_{p_+}[\log p_+] \quad (8.14)$$

$$= \text{entropy}(p) - \log Z[\Theta_+] - \sum_i (< \lambda_i^{(\beta)}, h_i >) - < \lambda_+^{(\beta)}, h_+ > \quad (8.15)$$

Taking the derivative with respect to h_+ gives

$$\frac{\partial D}{\partial h_+} = 0 - \frac{\partial \log Z[\Theta_+]}{\partial h_+} - \sum_i (< \frac{\partial \tilde{\lambda}_i^{(\beta)}}{\partial h_+}, h_i >) - < \frac{\partial \lambda_+^{(\beta)}}{\partial h_+}, h_+ > - \lambda_+^{(\beta)} h_+, \quad (8.16)$$

where $\tilde{\lambda}_i^{(\beta)}$ are the newly estimated λ parameters given the introduction of the new relationship histogram h_+ .

Using

$$\frac{1}{Z[\Theta_+]} \frac{\partial Z[\Theta_+]}{\partial h_+} = \frac{\partial \log Z[\Theta_+]}{\partial h_+}, \quad (8.17)$$

the first term in 8.16 cancels out and we are left with

$$\frac{\partial D}{\partial h_+} = -\lambda_+^{(\beta)} h_+. \quad (8.18)$$

Because this equation does not involve h_0 and the $\lambda_+^{(\beta)}$ parameters are independent of h_0 and

$$D'(h_0) = 0 . \quad (8.19)$$

Thus we can write our distance estimate for $KL(p_+||p)$ as

$$KL(p_+||p) = D(h_+) = \frac{1}{2}(h_+ - h_0)^T D''(h^*)(h_+ - h_0) . \quad (8.20)$$

Because $E_{p_+}[H^{r+}] = E_f[H^{r+}]$, we can estimate h_+ using $(H_{r_+}^{(\beta)}(PG^{obs})$ and h_0 using $H_{r_+}^{(\beta)}(PG^{syn})$. Thus,

$$D(h_+) = d_{mahn}(H_{r_+}^{(\beta)}(PG^{obs}), H_{r_+}^{(\beta)}(PG^{syn})) . \quad (8.21)$$

Appendix C

Our goal is to compute the proposal probability ratio $\frac{q(cccp_o|A)}{q(cccp_o|B)}$ for selecting a composite connected component $cccp_o$ at two states $X = A$ and $X = B$ which differ only in the labels of nodes in $cccp_o$.

At state A , step 1 of \mathcal{C}^4 samples all the edge variables $U = \{u_e, : e \in E\}$ independently following the Bernoulli probabilities. Let $E_{on}(U|A)$ and $E_{off}(U|A)$ denote all the edges that are turned on and off respectively which depend on state A . If $e \in E_{on}(U|A)$ then u_e was set to 'on' with probability q_e . If $e \in E_{off}$, it can be turned off either with probability $1 - q_e$ or deterministically (i.e. the nodes at the two ends have different labels for positive edges or the same label for negative edges). We are only concerned with those edges that are turned on or off probabilistically and denote it by

$$\begin{aligned} E_{off}^*(U|A) &= \{e : u_e = 'off', e \in E^+, x_i = x_j\} \\ &\cup \{e : u_e = 'off', e \in E^-, x_i \neq x_j\}. \end{aligned}$$

We disregard the edges that are turned off deterministically as the later are not relevant to the proposal probability. These edge variables U form a graph $g(A) = \langle V, E_{on} \rangle$ with probability

$$p(g(A)|A) = \prod_{e \in E_{on}(U|A)} q_e \prod_{e \in E_{off}^*(U|A)} (1 - q_e).$$

In $g(A)$, suppose we have a set of $cccp$'s, and denote them by $CP(U|A)$, and we pick up one as $cccp_o$ with uniform probability $\frac{1}{|CP(U|A)|}$. Given state A , there are many possible U 's that can generate a certain $cccp_o$ by altering the connections inside the $cccp$'s, we denote the set of such U 's by

$$\Omega(cccp_o|A) = \{U : cccp_o \in CP(U|A)\}.$$

Therefore, the probability for choosing a particular $cccp_o$ at state A is $q(cccp_o|A) =$

$$\sum_{U \in \Omega(cccp_o|A)} \frac{1}{|CP(U|A)|} \prod_{e \in E_{on}(U|A)} q_e \prod_{e \in E_{off}^*(U|A)} (1 - q_e)$$

All the U 's in $\Omega(cccp_o|A)$ share a common subset of edges $Cut(cccp_o|A) \subset E_{off}^*(U|A)$ which must be turned off probabilistically for $cccp_o$ being a valid composite connected component. Thus we rewrite

$$\begin{aligned} q(cccp_o|A) &= \left(\prod_{e \in Cut(cccp_o|A)} (1 - q_e) \right) \\ &\left[\sum_{U \in \Omega(cccp_o|A)} \frac{1}{|CP(U|A)|} \prod_{e \in E_{on}(U|A)} q_e \right. \\ &\quad \left. \prod_{e \in E_{off}^*(U|A) \setminus Cut(cccp_o|A)} (1 - q_e) \right]. \end{aligned}$$

Similarly, the probability for selecting $cccp_o$ at state B is

$$\begin{aligned} q(cccp_o|B) &= \left(\prod_{e \in Cut(cccp_o|B)} (1 - q_e) \right) \\ &\left[\sum_{U \in \Omega(cccp_o|B)} \frac{1}{|CP(U|B)|} \prod_{e \in E_{on}(U|B)} q_e \right. \\ &\quad \left. \prod_{e \in E_{off}^*(U|B) \setminus Cut(cccp_o|B)} (1 - q_e) \right]. \end{aligned}$$

As states A and B differ only in their labels of $cccp_o$, there is a one-to-one identical match between $g(A)$ and $g(B)$, and thus one-to-one match between $CP(U|A)$ and $CP(U|B)$. The only difference is the cuts $Cut(cccp_o|A) \neq Cut(cccp_o|B)$. All other terms are cancels, and we have

$$\frac{q(cccp_o|A)}{q(cccp_o|B)} = \frac{\prod_{e \in Cut(cccp_o|A)} (1 - q_e)}{\prod_{e \in Cut(cccp_o|B)} (1 - q_e)}.$$

BIBLIOGRAPHY

- Ali, S. and Shah, M. (2005). An integrated approach for generic object detection using kernel pca and boosting. *IEEE International Conference on Multimedia and Expo*. 9
- Apt, K. R. (1999). The essence of constraint propagation. *Theoretical Computer Science*. 74
- Barbu, A. and Zhu, S.-C. (2005). Generalizing swendsen-wang to sampling arbitrary posterior probabilities. *Pattern Analysis and Machine Intelligence*, 27:1239–1253. 72, 76, 86, 93
- Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. (2008). Surf: Speeded up robust features. *Computer Vision and Image Understanding*, 110(3):346–359. 10
- Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *Transactions on Pattern Analysis and Machine Intelligence*, 24(24):509–521. 53
- Berg, A., Grabler, F., and Malik, J. (2007). Parsing images of architectural scenes. *IEEE 11th International Conference on Computer Vision*. 17
- Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of Royal Statistical Society*, B-48(259):302. 72
- Blei, D. and Jordan, M. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, pages 993–1022. 10
- Boichis, N., Viglino, J.-M., and Cocquerez, J.-P. (2000). Knowledge based system for the automatic extraction of road intersections from aerial images. *International Archives of Photogrammetry and Remote Sensing*. 17

- Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239. [72](#), [76](#), [95](#)
- Cao, L. and Li, F.-F. (2007). Spatially coherent latent topic model. *International Conference on Computer Vision*. [15](#)
- Chen, H., Xu, Z., Liu, Z., and Zhu, S.-C. (2006). Composite templates for cloth modeling and sketching. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1:943–950. [13](#), [16](#)
- Chi, Z. and Geman, S. (1998). Estimation of probabilistic context-free grammars. *Computational Linguistics*, 24(2). [42](#)
- Chomsky, N. (1956). Three models of the description of language. *IRE Transactions on Information Theory*, (2):113–124. [19](#)
- Chui, H. and Rangarajan, A. (2003). A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89(2):114–141. [72](#), [80](#)
- Cooper, C. and Frieze, A. (1999). Mixing properties of the swendsen-wang process on classes of graphs. *Proceedings of DIMACS Workshop on Statistical Physics Methods in Discrete Probability, Combinatorics and Theoretical Computer Science*, 41:1499–1527. [76](#)
- Cormen, T., Leiserson, C., Rivest, R., and Stein, C. (1988). *Introduction to Algorithms, Second Edition*. MIT Press and McGraw-Hill. [72](#)
- Crandall, D., Felzenszwalb, P., and Huttenlocher, D. (2005). Spatial priors for part-based recognition using statistical models. *Computer Vision and Pattern Recognition*. [11](#)

- Dellaert, F., Seitz, S., and C. Thorpe, S. T. (2001). Feature correspondence: A markov chain monte carlo approach. *Advances in Neural Information Processing Systems*, 13:852–858. [76](#)
- Dickinson, S., Pentland, A., and Rosenfeld, A. (1992). From volume to views: an approach to 3d object recognition. *CVGIP Graphical Models and Image Processing: Image Understanding*, 55(2):130–154. [13](#), [14](#)
- Edwards, R. G. and Sokal, A. (1988). Generalization of the fortuin-kasteleyn-swendsen-wang representation and monte carlo algorithm. *Physical Review, the American Physical Society*. [82](#), [95](#)
- et al., A. G. (2004). *Bayesian Data Analysis, Second Edition*. Chapman and Hall CRC. [72](#)
- Felzenszwalb, P. and Huttenlocher, D. (2005). Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79. [11](#), [71](#)
- Fergus, R., Perona, P., and Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. *International Conference on Computer Vision*. [11](#), [12](#)
- Fergus, R., Perona, P., and Zisserman, A. (2005). A sparse object category model for efficient learning and exhaustive recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [11](#)
- Fischler, M. and Elschlager, R. (1973). The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 22(1):67–92. [11](#)
- Fletcher, R. (1970). A new approach to variable metric algorithms. *Computer Journal*. [100](#)

- Forssen, P. E. and Lowe, D. (2007). Shape descriptors for maximally stable extremal regions. *International Conference on Computer Vision*. 10
- Freund, Y. and Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*. 10, 17, 114, 115
- Fu, K.-S. (1981). *Syntactic Pattern Recognition and Applications*. Prentice Hall. 13
- Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence*. 72, 75, 95
- Green, P. (1995). Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82:711–732. 75
- Grenander, U. and Miler, M. (1994). Representations of knowledge in complex systems. *Journal of the Royal Statistical Society Series B*, 56(4). 75
- Han, F. and Zhu, S.-C. (2005). Bottom-up and top-down image parsing by attribute graph grammar. *Proceedings of the International Conference on Computer Vision*, 2. 13, 15
- Hinz, S. and Baumgartner, A. (2000). Road extraction in urban areas supported by context objects. *International Archives of Photogrammetry and Remote Sensing*, 33. 17
- Hoffman, T. (1999). Probabilistic latent semantic analysis. *Uncertainty in Artificial Intelligence*. 10
- Jin, Y. and Geman, S. (2006). Context and hierarchy in a probabilistic image model.

- Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2:2145–2152. [15](#)
- Keselman, Y. and Dickinson, S. (2001). Generic model abstraction from examples. *Pattern Analysis and Machine Intelligence*, 27:1141–1156. [13](#)
- Kolmogorov, V. and Rother, C. (2007). Minimizing nonsubmodular functions with graph cuts—a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(9):1274–1279. [72](#), [76](#)
- Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86. [45](#)
- Kumar, M. P. and Torr, P. (2006). Fast memory-efficient generalized belief propagation. *Lecture Notes in CS*. [72](#), [95](#)
- Kumar, S. and Hebert, M. (2003). Man-made structure detection in natural images using a causal multiscale random field. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. [71](#), [72](#), [100](#), [101](#), [102](#)
- Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of the Eighteenth International Conference on Machine Learning*. [72](#)
- Li, F.-F. and Perona, P. (2005). A bayesian hierarchical model for learning natural scene categories. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2:524–531. [15](#)
- Li, Y., Atmosukarto, I., Kobashi, M., Yuen, J., and Shapiro, L. G. (2005). Object and event recognition for aerial surveillance. *SPIE - The International Society for Optical Engineering*. [17](#)

- Lin, L., Zeng, K., Liu, X. B., and Zhu, S.-C. (2009). Layered graph matching by composite clustering with collaborative and competitive interactions. *IEEE Proceedings of the Conference on Computer Vision and Pattern Recognition*. 72
- Liu, J. (2001). *Monte Carlo Strategies in Scientific Computing*. Springer. 75
- Liu, J., Wong, W., and Kong, A. (1995). Correlation structure and convergence rate of the gibbs sampler with various scans. *The Journal of the Royal Statistical Society*, 57(1):157–169. 72, 75
- Liu, W. and Prinet, V. (2005). Building detection from high-resolution satellite image using probability model. *Geoscience and Remote Sensing Symposium, IGARSS*, pages 25–29. 17
- Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110. 10
- Mackworth, A. K. (1973). Interpreting pictures of polyhedral scenes. *Artificial Intelligence*, 4(2):121–137. 83
- Mackworth, A. K. (1977). Consistency in networks of relations. *Artificial Intelligence*, pages 99–118. 74
- Maloof, M., Langley, P., Binford, T., Nevatia, R., and Sage, S. (2003). Improved rooftop detection in aerial images with machine learning. *Machine Learning*. 17
- Matsuyama, T. and Hang, V. (1990). Sigma: A framework for image understanding integration of bottom-up and top-down analyses. *Plenum*. 17
- Moghaddam, B. (1999). Principal manifolds and bayesian subspaces for visual recognition. *International Conference on Computer Vision*. 9

- Moissinac, H., Maitre, H., and Bloch, I. (1994). Urban aerial image understanding using symbolic data. *Image and Signal Processing for Remote Sensing, Proc. SPIE*. 17
- Nayar, S. K., Murase, H., and Nene, S. A. (1996). *Parametric Appearance Representation*. 8
- Oh, S. M., Rehg, J., Balch, T., and Dellaert, F. (2005). Learning and inference in parametric switching linear dynamical systems. *IEEE International Conference on Computer Vision*, 2:1161–1168. 76
- Pearl, J. (1984). *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Longman Publishing Co., Inc. 14, 74, 75
- Porway, J., Wang, K., Yao, B., and Zhu, S.-C. (2008). A hierarchical and contextual model for aerial image understanding. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 71
- Rosenfeld, A., Hummel, R. A., and Zucker, S. W. (1976). Scene labeling by relaxation operations. *IEEE Trans. on SMC*, (6):420–433. 71, 74
- Shotton, J., Winn, J., Rother, C., and Criminisi, A. (2006). Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. *Proceedings of the European Conference on Computer Vision*, pages 1–15. 17, 115
- Siddiqi, K., Shokoufandeh, A., Dickinson, S., and Zucker, S. W. (1999). Shock graphs and shape matching. *International Journal of Computer Vision*, 35(1):13–32. 13
- Singhal, A., Luo, J., and Zhu, W. (2003). Probabilistic spatial context models for scene content understanding. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1. 15

- Sivic, J., Russell, B., Efros, A., Zisserman, A., and Freeman, W. (2005). Discovering objects and their location in images. *Tenth IEEE International Conference on Computer Vision*. 15, 17
- Sudderth, E. B., Torralba, A., Freeman, W. T., and Willsky, A. S. (2005). Describing visual scenes using transformed dirichlet processes. *Neural Information Processing Systems*. 15
- Sugihara, K. (1986). *Machine Interpretation of Line Drawings*. MIT Press. 83, 99
- Swendsen, R. and Wang, J.-S. (1987). Nonuniversal critical dynamics in monte carlo simulations. *Physicl Review Letters*. 72, 75, 82, 86, 95
- Todorovic, S. and Ahuja, N. (2006). Extracting subimages of an unknown category from a set of images. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1:927–934. 14
- Torralba, A., Murphy, K., and Freeman, W. (2004). Sharing visual features for multiclass and multiview object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 15, 72
- Tu, Z. and Zhu, S. C. (2002). Image segmentation by data-driven markov chain monte carlo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 72, 76
- Turk, M. and Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):585–591. 9
- Ullman, S., Sali, E., and Vidal, M. (2001). A fragment-based approach to object representation and classification. *Proceedings of the 4th International Workshop on Visual Form*. 10

- Vestri, C. (2001). Using robust methods for automatic extraction of buildings. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1. 17
- Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 511–518. 17
- Weber, M., Welling, M., and Perona, P. (2000). Towards automatic discovery of object categories. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2:101–108. 10
- Weiss, Y. (2000). Neural computation. *Correctness of Local Probability Propagation in Graphical Models with Loops*, 12(1). 72, 76
- Winn, J. and Shotton, J. (2006). The layout consistent random field for recognizing and segmenting partially occluded objects. *Computer Vision and Pattern Recognition*. 14
- Wu, T. F., Xia, G.-S., and Zhu, S.-C. (2007). Compositional boosting for computing hierarchical image structures. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. 116
- Wu, T.-F. and Zhu, S.-C. (2010). A numeric study of the bottom-up and top-down inference processes in and-or graphs. *International Journal of Computer Vision*. 105
- Yao, B., Yang, X., and Zhu, S.-C. (2007). Introduction to a large scale general purpose ground truth dataset: Methodology, annotation tool, and benchmarks. *Energy Minimization Methods in Computer Vision and Pattern Recognition*, 4697:169–183. xii, 62

- Zhang, J., Marszalek, M., Lazebnik, S., and Schmid, C. (2001). Local features and kernels for classification of texture and object categories: a comprehensive study. *International Journal of Computer Vision*, 73(2):213–238. [10](#)
- Zhao, T. and Nevatia, R. (2001). Car detection in low resolution aerial images. *IEEE International Conference on Computer Vision*, 1. [17](#)
- Zhu, L., Lin, C., Huang, H., Chen, Y., and Yuille, A. (2008). Unsupervised structure learning: Hierarchical recursive composition, suspicious coincidence and competitive exclusion. *Proceedings of the 10th European Conference on Computer Vision: Part II*. [15](#)
- Zhu, S.-C. and Mumford, D. (2006). A stochastic grammar of images. *Foundation and Trends in Computer Graphics and Vision*, 2(4):259–362. [72](#)
- Zhu, S.-C., Wu, Y.-N., and Mumford, D. (1997). Minimax entropy principle and its application to texture modeling. *Neural Computation*, 9(9):1627–1660. [133](#)